НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

Дипломна робота

на тему: Модуль «Студент» системи управління дипломними проектами

Студент групи ТВ-51 Чехівський Андрій Миколайович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник роботи доц., к.т.н. Коваль Олександр Васильович _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Кількість сторінок 75

Кількість ілюстрацій 30

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050103 “Програмна інженерія”

на тему: Модуль “Студент” системи управління дипломними проектами

Виконав: студент 4 курсу, групи ТВ-51

Чехівський Андрій Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц., к.т.н Коваль Олександр Васильович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050103 “Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Чехівському Андрію Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Модуль “Студент” системи управління дипломними проектами”
керівник роботи доц., к.т.н Коваль Олександр Васильович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)
затверджена наказом вищого навчального закладу від ”___”_____ 2019__ р. № _____ __

2. Строк подання студентом роботи 10 червня 2019__ р.
3. Вихідні дані до роботи React.js, веб-застосунок
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби управління проектами, спроектувати архітектуру системи, розробити програмне забезпечення, реалізувати веб-застосунок.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень) Мета та завдання роботи, Функції системи, Діаграма прецедентів, Будова системи, Використані засоби розробки, Методика роботи користувача з системою, Висновки

Дата видачі завдання "_10_" __жовтня__ 2019__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.18-15.01.19	
2.	Розробка архітектури та загальної структури системи	15.01.19-04.02.19	
3.	Розробка структур окремих підсистем	05.02.19-11.02.19	
4.	Підготовка матеріалів	12.02.19-15.02.19	
5.	Програмна реалізація системи	16.02.19-28.02.19	
6.	Захист програмного продукту	15.05.19	
7.	Оформлення пояснювальної записки	05.03.19-29.05.19	
8.	Передзахист	28.05.19	
9.	Захист		

Студент _____ Чехівський А.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Коваль О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи було створення модулю “Студент” системи управління дипломними проектами, що надає можливість покращити комунікацію між студентами та викладачами і підвищити ефективність людських ресурсів.

Розроблений програмний продукт представляє собою веб-застосунок і дозволяє користувачам шукати та переглядати теми дипломних робіт, подавати і відміняти заявки на теми дипломних робіт, переглядати інформацію по своїй дипломній роботі, переглядати навантаження на викладачів.

Програмний продукт реалізовано в середовищі програмування WebStorm за допомогою мови програмування JavaScript. База даних була створена у об’єктно-реляційній системі керування базами даних MySQL.

Записка містить 75 сторінок, 30 рисунків та 25 посилань.

ABSTRACT

The purpose of the work was to create a module “Student” of the diploma project management system, which provides an opportunity to improve communication between students and teachers and increase the efficiency of human resources.

The developed software is a web-based application and allows users to search and browse the topics of theses, submit and cancel applications for topics of theses, review information on their graduation thesis, review the load on teachers.

The software product is implemented in the WebStorm programming environment using the JavaScript programming language. The database was created in the object-relational database management system MySQL.

The note contains 75 pages, 30 images and 25 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних.

ВСТУП

Управління проектом — це методологія організації, планування, керівництва, координації трудових, фінансових та матеріально-технічних ресурсів протягом проектного циклу, спрямована на ефективне досягнення його цілей шляхом застосування сучасних методів, техніки та технології управління для досягнення певних результатів, щодо складу та обсягу робіт, вартості, часу, якості та задоволення учасників проекту.

Однією з актуальних проблем навчального процесу в університеті є управління дипломними проектами. Наприклад, студенту необхідно знаходи інформацію про графік виконання дипломного проекту, доступні теми, список дипломних керівників.

Внаслідок цього виникають певні проблемні ситуації, коли студент не знає до кого звернутися за інформацією, як знайти викладача, невчасно або неякісно виконана робота.

В еру цифрових технологій, коли електронні пристрої майже завжди знаходяться поруч, значно легше контролювати та коригувати дипломні роботи за допомогою спеціальних програм. Особливої важливості набуває Система управління дипломними проектами, метою якої є створення інструменту здатного підвищити ефективність людських ресурсів.

Перший розділ містить постановку задачі даної дипломної роботи. Другий розділ містить опис існуючих програмних рішень та переваги розроблюваної системи перед ними. В третьому розділі описуються інструменти та методи, що використовувались для виконання дипломної роботи. Четвертий розділ містить опис програмної реалізації всієї системи. П’ятий розділ описує методику роботи користувача з розробленим програмним продуктом. Далі знаходяться висновки, що були зроблені під час виконання.

1. ПОСТАНОВКА ЗАДАЧІ РЕАЛІЗАЦІЇ МОДУЛЯ СТУДЕНТ СИСТЕМИ УПРАВЛІННЯ ДИПЛОМНИМИ ПРОЕКТАМИ

Метою роботи є розробка модуля “Студент” системи управління дипломними проектами, що надає можливість підвищити ефективність людських ресурсів і покращити комунікацію між викладачами та студентами.

Розроблена система повинна бути представленна як веб-застосунок.

Задачами, які мають виконуватись даним продуктом є:

— створення серверної частини, що має змогу взаємодіяти з базою даних;

— база даних, що задовольняє тему дипломної роботи;

— розробка веб-застосунку.

Система має забезпечувати наступні можливості:

— шукати та переглядати теми дипломних робіт;

— подавати і відміняти заявки на теми дипломних робіт;

— переглядати інформацію по своїй дипломній роботі після прийняття заявки викладачем;

— переглядати графік виконання дипломної роботи;

— переглядати навантаження на викладачів;

— переглядати інформацію про склад та напрямки лабораторії;

— переглядати та редагувати власний профіль;

— вхід в систему;

— відновлення паролю.

2. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

У даній роботі було проведено аналіз проблеми управління дипломними проектами та знайдено рішення для полегшення роботи над дипломним проектом.

Дипломний проект — це робота, яка повинна об’єктивно оцінити уміння вирішувати типові задачі, які спрямовані в освітньо-кваліфікаційних характеристиках до проектної і виконавської робочим функціям[1].

Дипломні проекти (роботи) виконуються на кінцевому етапі навчання студентів і передбачають: закріплення, систематизацію та розширення знань, як практичних, так і теоретичних зі своєї спеціальності та вміння застосувати їх при вирішенні конкретних технічних, наукових, економічних, виробничих і інших завданнь; розвиток навичок самостійної роботи й оволодіння методикою дослідження та експеримент, пов’язаних з темою проекту (роботи).

2.1.Застосунок Trello

Застосунок Trello — це одна з найпростіших і зручних систем управління проектами. Вона універсальна, легка, гнучка, з відкритим API, а головне, безкоштовна. Її найсильніші сторони: можливість паралельно відстежувати статус різних завдань на одному екрані і зручна інтеграція з іншими популярними інструментами. Це максимально простий інструмент, який легко впровадити в робочий процес без довгої адаптації з боку персоналу.

Весь інтерфейс збудований на основі канбан-дощок. Для організації завдань використовується дошка з картками, які розподіляються за типами. Як правило, завдання розбиваються на заплановані, поточні і виконані.

Якщо ви берете участь у великій кількості робочих проектів або якщо ви самозайняті, то ефективне управління проектами багато. Якщо ви не плануєте свій

час і зусилля, то в кінцевому підсумку ви знехтуйте його на Facebook і в результаті відсутні терміни. Індустрія інструментів онлайн-управління проектами є дуже конкурентоспроможною, але, як здається, вона перемагає над усіма іншими, це Trello (рисунок 2.1).

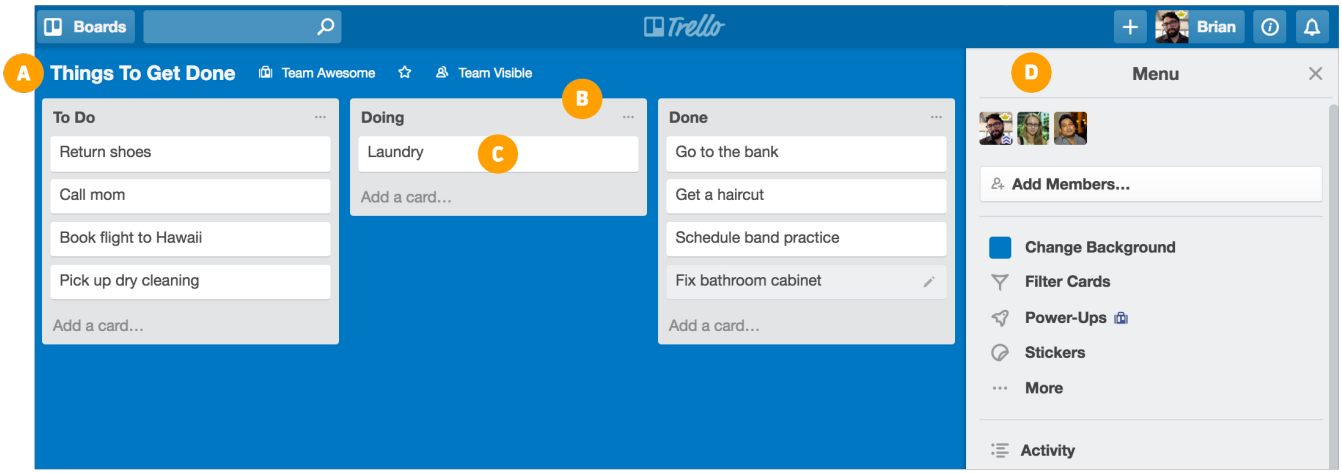

Рисунок 2.1 — Зразок інтерфейсу сайту Trello

Trello улюблений тими, хто одержимий управлінням своїм часом і проектами, і легко зрозуміти, чому. Trello не тільки дешевий і доступний, але й надзвичайно простий у використанні. Використовуючи дуже візуальний підхід до управління часом і проектами, ви можете переміщати карти навколо, щоб ви могли побачити, що потрібно зробити і коли.

Trello — це інструмент спільної роботи, який організовує ваші проекти в дошки. Одним поглядом, Трелло розповідає вам про те, на чому працює, хто працює над чим і де щось знаходиться в процесі.

Уявіть собі білу дошку, заповнену списками наліпок, кожна нота як завдання для вас і вашої команди. Тепер уявіть, що кожна з цих наліпок містить фотографії, вкладення з інших джерел даних, таких як BitBucket або Salesforce, документи, і місце для коментарів і співпраці з вашими товаришами по команді. А тепер уявіть, що ви можете взяти цю дошку в будь-якому місці вашого смартфона і отримати доступ до неї з будь-якого комп’ютера через Інтернет.

З мінусів я можу виділити те, що:

— функціоналу Trello недостатньо для дійсно великих компаній;

— на маленьких екранах Trello стає не такою зручною;

— автоматизацію, повторення і швидке додавання завдань важкувато реалізувати.

Незважаючи на деякі недоліки, Trello нарощує популярність і активно впроваджується компаніями в самих різних сферах. Це прекрасне онлайн-рішення для швидкої і легкої організації роботи.

Стан проектів в поточний момент часу і можливість запускати кілька проектів одночасно є головною перевагою Trello. Ця система надає можливість контролювати виконання проектів в будь-який момент часу і бачити зміни в режимі реального часу.

Ця система підходить для домашніх проектів, просто наборів ідей і стартапів, але не для великих проектів.

2.2.Застосунок Atlassian JIRA

Застосунок JIRA — це система, яка підходить для стеження за вадами і управління проектом в компанії будь-якого розміру. Це інструмент для всіх співробітників в команді і керівників проектів.

JIRA допомагає команді обмінюватися інформацією і легко залучати різних співробітників в проекти і завдання, відстежувати і фіксувати помилки користувачів в роботі з програмними продуктами, забезпечувати дотримання роботи точно в строк і в рамках регламенту робочого процесу, перевіряти і планувати ефективність працівників і призначати певні завдання, працювати разом з колегами за допомогою інструментів спільного редагування файлів, а також відстежувати прогрес і оновлення кожного завдання команди.

Динамічні інструменти системи для управління проектами JIRA дають можливість керівникам виявити перешкоди, які не дають команді працювати ефективніше, приймати цілеспрямовані дії по їх усуненню і визначати області поліпшення робочого процесу. Необхідно багато часу і зусиль, щоб з’ясувати, як ефективно використовувати цей інтерфейс.

Основні можливості JIRA

— функціоналу Trello недостатньо для дійсно великих компаній;
— перегляд процесу роботи розробників над проектом, які виникають помилки в його використанні клієнтами, швидке їх усунення;
— документи, пошта, спілкування знаходяться в одному місці, рішення кожного питання відбувається швидко і легко;
— планування робочого процесу;
— постановка завдань, пріоритетів, допомога команді у виділенні важливого в роботі і відстеження виконання завдань;
— обмін інформацією по проекту, спільне вирішення питань і звернення за допомогою до колег;
— новини, пошта, перегляд роботи в режимі реального часу в одному місці;
— інтеграція з різними розробками і доповненнями Altassian і іншими розробниками.

Завдяки можливості настроювати JIRA її можна застосовувати і для задач поза IT, зокрема для управління HR, для ризик-менеджменту і управління вимогами.

Для виклику віддалених процедур використовується REST, раніше також була підтримка SOAP і XML-RPC, але від них відмовились у версії 7.0.[2] В JIRA є підтримка англійської, французької, японської, іспанської і німецької мов. За допомогою додатків також можна добавити китайську, чеську, данську, італійську, норвезьку, польську, португальську, російську і словацьку мови[3].

JIRA відмінно підходить для швидких команд розробників і користувачів, які люблять технічне, складне програмне забезпечення.

Питання JIRA починаються з застарілого інтерфейсу. Програма має багато неефективності і часто виявляється надмірно складним. Команди управління проектами, які спробували параметри налаштування, вважають цей процес громіздким. Ті, хто шукає спосіб підвищити продуктивність, знайдуть дизайн дизайну JIRA користувачем невиправдано складним.

Зрештою, ніхто не буде сперечатися з тим, що JIRA була побудована на увазі команд розвитку. Організації, які не покладаються на розробників, знайдуть JIRA неефективною.

JIRA - це інструмент для відстеження випуску та проекту Atlassian. Простіше кажучи: JIRA дозволяє відстежувати будь-яку одиницю роботи (будь то проблема, помилка, історія, завдання проекту тощо) через заздалегідь визначений робочий процес (рисунок 2.2).

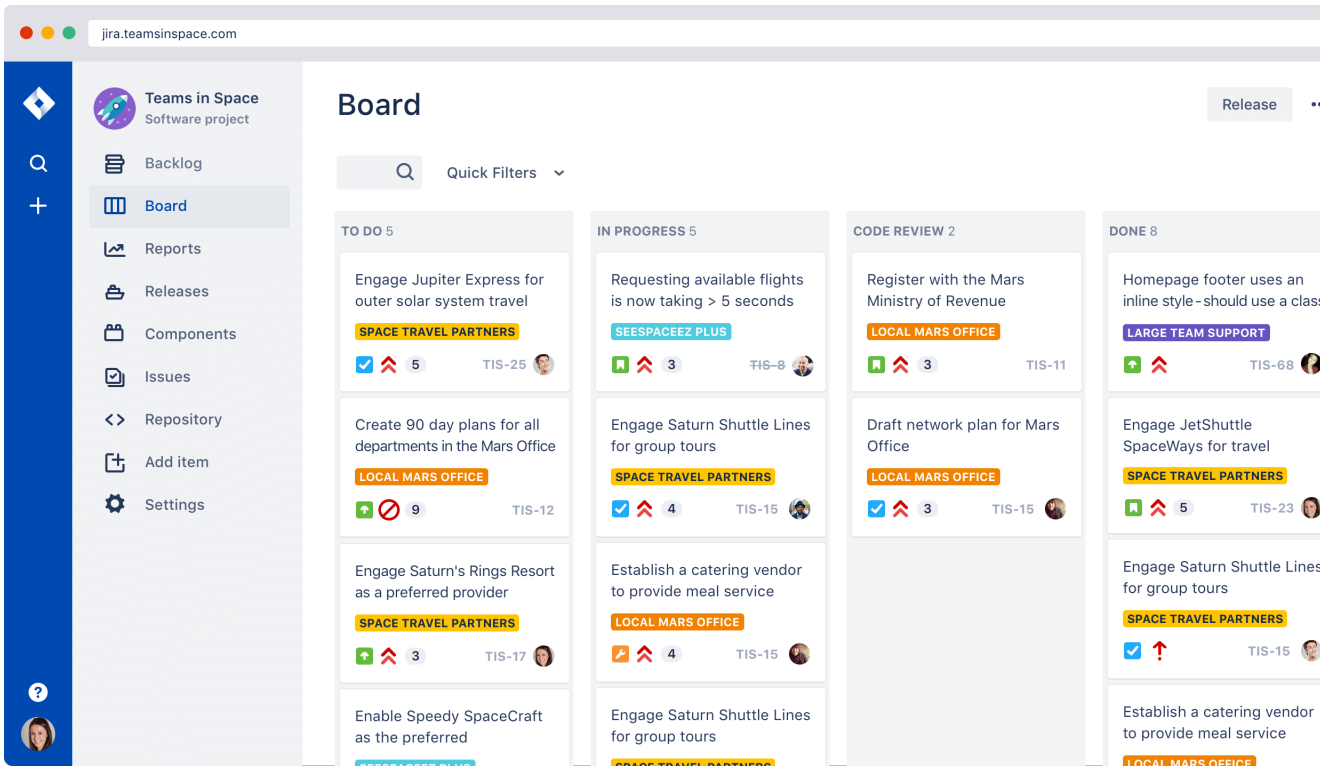

Рисунок 2.2 — Зразок інтерфейсу сайту Jira

Обидва пристрої (те, що ви називаєте робочим елементом) і робочий процес (кроки, які виконується елементом з відкритих до закритих), можуть бути налаштовані спеціально для конкретних вимог вашої команди, будь то прості або складніші.

Крім того, JIRA дійсно добре відстежує (через детальні, спеціальні звіти та інформаційні панелі), де всі ваші проекти / питання лежать на рівні команди, компанії або особистості - наприклад, які питання стосуються мене , створених за останні 7 днів?

Деякі поширені випадки використання включають розробку програмного забезпечення, реалізацію функцій, відстеження помилок, гнучке управління проектами (з JIRA Agile) та відстеження квитка на службу (з JIRA Service Desk).

Jira має високу конфігурацію та гнучкість, що дозволяє використовувати їх у різних середовищах та процесах. Робочі процеси Jira, типи випусків і екрани дозволяють пристосовувати практично будь-який сценарій і легко можуть бути змінені за допомогою графічного інтерфейсу адміністратора. Atlassian надає відмінні ресурси онлайн-підтримки та особисто групи користувачів Atlassian, щоб допомогти спільноті.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Дану систему було написано на мові Javascript в операційній системі macOS. Середовищем розробки був обраний WebStorm та використанням бібліотеки React.js та фреймворку Express.js.

Інтерфейс користувача було реалізовано за допомогою бібліотеки React.js на мові JavaScript, які стилізуються за допомогою бібліотеки styled-components.

Серверну частину було реалізовано за допомогою фреймворку Express.js на мові JavaScript.

Дані про дипломні роботи зберігаються у СКБД MySQL.

3.1.Вибір архітектури системи

Розроблювана система повинна мати клієнт-серверну архітектуру [4]. Це мережева або обчислювальна архітектура, в середині якої, завдання розподілені між постачальником послуг (функцій), що називається сервером, і замовниками, що називаються клієнтами. Загалом, клієнт та сервер являють собою програмне забезпечення. Зазвичай ці програми розташовані на різних обчислювальних машинах і взаємодіють між собою через обчислювальну мережу за допомогою мережевих протоколів, але вони можуть бути розташовані також і на одній машині.

Архітектура клієнт-сервер — це обчислювальна модель, в якій сервер розміщує, доставляє і управляє більшістю ресурсів і послуг, які споживається клієнтом. Цей тип архітектури має один або більше клієнтських комп’ютерів, підключених до центрального сервера через мережу або підключення до Інтернету.

Архітектура клієнт-сервер також відома як мережева обчислювальна модель або мережа клієнn-сервер, оскільки всі запити та послуги доставляються через мережу.

Архітектура клієнт-сервер — це архітектура виробників та споживачів, де сервер виступає як виробник і клієнт як споживач (рисунок 3.1). Сервер розміщує і надає клієнтові послуги високого класу на вимогу. Ці послуги можуть включати доступ, зберігання, спільний доступ до файлів, доступ до принтера та, або прямий доступ до необробленої обчислювальної потужності сервера.

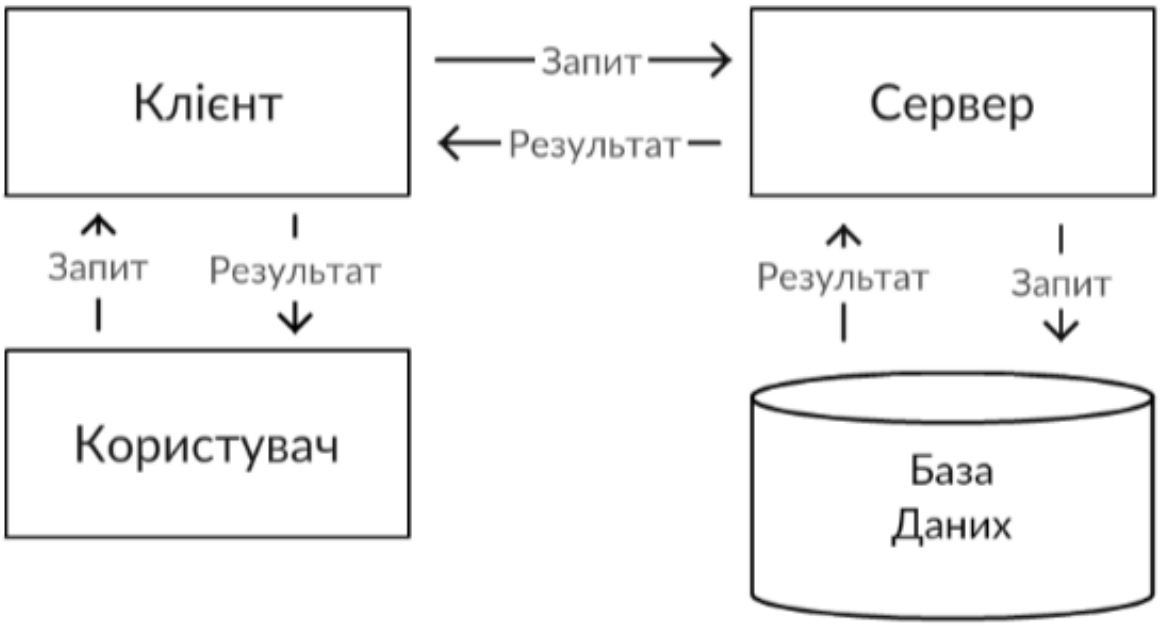

Рисунок 3.1 — Схематичне зображення клієнт – серверної архітектури

Архітектура клієнт-сервер працює, коли клієнтський комп’ютер посилає запит ресурсу або процесу на сервер через мережеве з’єднання, яке потім обробляється і доставляється клієнту. Серверний комп’ютер може керувати декількома клієнтами одночасно, тоді як один клієнт може бути підключений до декількох серверів одночасно, кожен з яких надає інший набір послуг. У найпростішій формі Інтернет також заснований на архітектурі клієнт-сервер, де веб-сервери обслуговують багатьох одночасних користувачів з даними веб-сайту.

Підвищення впливу моделі клієнт-сервер на вдосконалення індустрії онлайн породило вимоги клієнт-серверних додатків. Для спілкування користувачів з онлайновими бізнес-організаціями, що розповсюджуються через Інтернет, клієнт-серверні додатки відіграють значну роль. Тут важливість клієнт-серверної архітектури набуває важливого значення.

Архітектура клієнт-сервер — це спільна система архітектури, де завантажуються клієнт-сервер. Архітектура клієнт-сервер — це централізована система ресурсів, де сервер зберігає всі ресурси. Сервер отримує численні виступи на своєму краю для спільного використання ресурсів своїм клієнтам на запит. Клієнт і сервер можуть бути на одній або в мережі. Сервер глибоко стабільний і масштабований, щоб повертати клієнтам відповіді. Ця архітектура орієнтована на послуги, що означає, що обслуговування клієнтів не буде перервано. Архітектура клієнт-сервер підпорядковує мережевому трафіку, реагуючи на запити клієнтів, а не на повну передачу файлів. Він відновлює файловий сервер сервером баз даних.

Клієнтські комп'ютери реалізують зв'язок, щоб дозволити користувачеві комп'ютера запитувати послуги сервера і представляти результати, які повертає сервер. Сервери чекають появи запитів від клієнтів і повертають їх. Сервер, як правило, надає клієнтам стандартизований простий інтерфейс, щоб уникнути плутанини апаратного та програмного забезпечення. Клієнти розташовані на робочих місцях або на особистих машинах, в той же час сервери будуть розташовані десь потужні в мережі. Ця архітектура корисна в основному, коли клієнти та сервер мають окремі завдання, які вони виконують. Багато клієнтів можуть одночасно отримувати інформацію про сервер, а також клієнтський комп'ютер може виконувати інші завдання, наприклад, надсилати електронні листи.

3.2.Пакетний менеджер npm

Пакетний менеджер npm — менеджер пакетів для Node.js з сотнями тисяч пакетів. Незважаючи на те, що він створює деякі з ваших структур / організацій, це не є головною метою. Головною метою, як ви торкнулися, є автоматизоване управління залежностями та пакетами. Це означає, що ви можете вказати всі залежності вашого проекту у файлі package.json, тоді будь-який час, коли ви (або хтось інший) має почати роботу з проектом, вони можуть просто запустити npm

install і негайно мати всі встановлені залежності. Крім того, можна також вказати, на які версії залежить ваш проект, щоб запобігти розриву вашого проекту.

Це, безумовно, можна вручну завантажити бібліотеки, скопіювати їх у правильні каталоги, і використовувати їх таким чином. Проте, як ваш проект (і список залежностей) зростає, це швидко стає трудомістким і брудним. Це також ускладнює співпрацю та обмін вашим проектом.

Сподіваюся, це зробить більш зрозумілим, що є метою npm. Як розробник Javascript (як на стороні клієнта, так і на стороні сервера), npm є незамінним інструментом у моєму робочому процесі.

3.3.Середовище розробки WebStorm

Середовище розробки WebStorm (рисунок 3.2) — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA, як висококласний IDE для проектів веб-розробки.

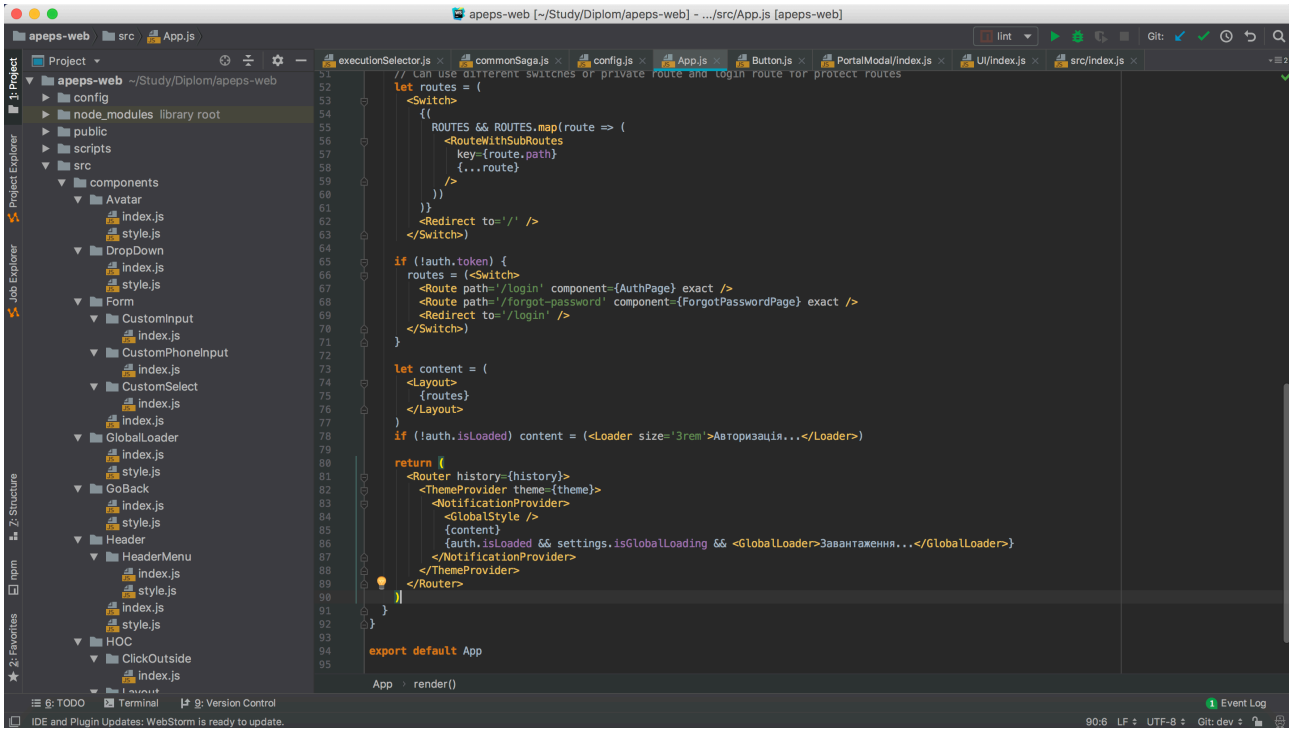

Рисунок 3.2 — Зразок інтерфейсу сайту середовища розробки WebStorm

WebStorm забезпечує інтуїтивну підтримку кодування для JavaScript та компільованих мов JavaScript, Node.js, HTML і CSS. Розробники можуть насолоджуватися кодовим завершенням, розширеними функціями навігації, розпізнаванням помилок на льоту та рефакторингами для всіх цих мов. Програма надає розширену допомогу для кодування Angular, React, Vue.js і Meteor. WebStorm також підтримує React Native, PhoneGap, Cordova і Ionic для розробки мобільних пристроїв і розробляє для серверної частини Node.js.

З усіма цими можливостями, доступними тільки в одній системі, розробникам не потрібно купувати окремі програми.

Інтуїтивно зрозумілий редактор WebStorm може ретельно оцінити проект, щоб отримати найкращі результати завершення коду для всіх підтримуваних мов. Сотні інтегрованих перевірок позначають будь-які можливі питання, що стосуються типів користувачів, а потім пропонують варіанти швидкого виправлення. Це забезпечує якість розробленого проекту.

Середовище розробки WebStorm має функції навігації та пошуку, які допомагають користувачам ефективніше та швидше обходити код. Можна перейти до визначення методу, функції або змінної. Це особливо корисно для розробників, які працюють над великими проектами.

Середовище розробки WebStorm може похвалитися вбудованими функціями для налагодження, тестування та відстеження ваших програм на стороні клієнта та Node.js. Завдання набагато простіше, оскільки вони вимагають лише мінімальної конфігурації. Користувачі можуть вказувати точки зупину, крокувати по коду і оцінювати вирази — все це не залишаючи IDE. WebStorm також має вбудований інструмент, який допомагає відслідковувати JavaScript коди. Файли пов’язані з викликами функцій і ідентифіковані потенційні зони проблем, щоб підвищити ефективність.

Середовище розробки WebStorm легко інтегрується з провідними інструментами командного рядка для веб-розробки. Це забезпечує оптимізований досвід розробки, що підвищує продуктивність без використання командного рядка. Інструменти збирання відображаються в простому уніфікованому інтерфейсі для виконання завдань Grunt, Gulp або npm.

Крім сотень власних перевірок WebStorm, технологія може запускати ESLint, JSCS, TSLint, Stylelint, JSHint або JSLint проти вашого коду. WebStorm має функцію місцевої історії, яка може перевіряти всі дії та повертатися до попередніх версій.

Розробники можуть працювати над більшою кількістю програм за допомогою вбудованих шаблонів WebStorm. Крім популярних, таких як Express або Web Starter Kit, користувачі можуть отримати доступ до більшої кількості генераторів проектів за допомогою інтеграції WebStorm з Yeoman.

Середовище розробки WebStorm дуже налаштовується, оскільки його можна налаштувати для будь-якого типу стилю кодування розробника. Користувачі можуть налаштовувати ярлики, шрифти та візуальні теми на вікно інструментів і макет редактора.

Основними перевагами цієї інтегрованої системи розробки є: рефакторинг, навігація по коду, автодоповнення, аналіз коду на льоту, миттєва перевірка змін через вплив інтеграцію з системами контролю версій.

Завдяки вбудованим інструментам тестування інтегрованої системи розробки, WebStorm можна виконувати тестування з найменшою кількістю зусиль та часу[5]. Програмісти можуть запускати і переглядати результати тестів у безпосередньо в середовищі розробки, перевірити код за допомогою програми Jest і протестувати React.js з Enzyme.

3.4.Мова програмування JavaScript

JavaScript є динамічною мовою комп’ютерного програмування[6]. Він легкий і найчастіше використовується як частина веб-сторінок, реалізація яких дозволяє клієнтському сценарію взаємодіяти з користувачем і створювати динамічні сторінки. Це інтерпретований мова програмування з об’єктно-орієнтованими можливостями.

JavaScript вперше був відомий як LiveScript, але Netscape змінив назву на JavaScript, можливо, через збудження Java. JavaScript вперше з’явився в Netscape 2.0 у 1995 році з назвою LiveScript[7]. Ядро мови загального призначення вбудовано в Netscape, Internet Explorer та інші веб-браузери.

Специфікація ECMA-262 визначила стандартну версію основної мови JavaScript:

— функціоналу Trello недостатньо для дійсно великих компаній;

— JavaScript є легкою, інтерпретованою мовою програмування;

— призначений для створення мережевих орієнтованих додатків;

— доповнює та інтегрується з Java;

— доповнює і інтегрується з HTML;

— відкрита і крос-платформна.

3.4.1.JavaScript на стороні клієнта

На стороні клієнта JavaScript є найпоширенішою формою мови. Сценарій повинен бути включений в HTML документ або на нього посилатися, щоб код був інтерпретований браузером.

Це означає, що веб-сторінка не повинна бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером і динамічно створюють вміст HTML.

Механізм на стороні клієнта JavaScript надає багато переваг перед традиційними сценаріями на стороні сервера CGI. Наприклад, ви можете скористатися JavaScript, щоб перевірити, чи введе користувач дійсну адресу електронної пошти у полі форми[8].

Код JavaScript виконується, коли користувач подає форму, і тільки якщо всі записи дійсні, вони будуть подані на веб-сервер.

JavaScript може використовуватися для уловлювання ініційованих користувачем подій, таких як кліки кнопок, навігація посилань та інші дії, які користувач ініціює явно або неявно[9].

3.4.2.Переваги JavaScript

Переваги використання JavaScript:

— менша взаємодія з сервером — ви можете перевірити вхід користувача перед відправкою сторінки на сервер, що економить навантаження на сервер;

— зворотний зв’язок з користувачами, яким не потрібно чекати, поки сторінка перезавантажиться, щоб побачити, що вони мають змогу щось ввести;
— підвищена інтерактивність — ви можете створювати інтерфейси, які реагують, коли користувач зависає над ними за допомогою миші або активує їх за допомогою клавіатури;
— більш чудові інтерфейси — ви можете використовувати JavaScript, щоб включити такі елементи, як компоненти перетягування і повзунки, щоб надати відвідувачам свого сайту багатий інтерфейс.

3.4.3.Обмеження JavaScript

Ми не можемо розглядати JavaScript як повноцінну мову програмування[10]. Вона не має таких важливих функцій:

— на стороні клієнта JavaScript не дозволяє читати або писати файли. Це зберігається з міркувань безпеки;
— JavaScript не можна використовувати для мережевих програм, оскільки такої підтримки немає;
— у JavaScript відсутні багатопоточні або багатопроцесорні можливості;
— знову ж таки, JavaScript є легкою, інтерпретованою мовою програмування, яка дозволяє створювати інтерактивність у статичних HTML-сторінках.

3.5.Бібліотека реалізації веб-інтерфейсу React.js

Бібліотека React.js — це бібліотека JavaScript з відкритим вихідним кодом, яка використовується для створення користувацьких інтерфейсів спеціально для односторінкових додатків. Він використовується для обробки шару уявлення для веб-додатків і мобільних додатків[11]. React також дозволяє нам створювати повторно використовувані компоненти користувальницького інтерфейсу. React був вперше створений Джорданом Уолке, інженером-програмістом, працюючим в Facebook. React вперше був розміщений на новинній стрічці Facebook в 2011 році і на Instagram.com в 2012 році.

React надає можливість розробникам створювати великі веб-додатки, які можуть змінювати дані без перезавантаження сторінки[12]. Основна мета React — бути швидким, таким, що масштабується і простим. Працює тільки на призначених для користувача інтерфейсів в додатку. Це відповідає уявленню в шаблоні MVC. Він може використовуватися з комбінацією інших бібліотек JavaScript або середовищ, таких як Angular JS в MVC.

3.5.1. Чому React.js?

Тепер, головне питання, яке виникає перед нами, це те, чому слід використовувати ReactJS. Існує дуже багато платформ з відкритим кодом для полегшення розробки веб-додатків, таких як Angular. Давайте швидко розглянемо переваги React у порівнянні з іншими конкурентними технологіями або структурами. Зі зміною фронт-енду на щоденній основі важко присвятити час вивченню нових рамок, особливо коли ці рамки в кінцевому підсумку можуть стати в глухий кут. Отже, якщо ви шукаєте наступну кращу річ, але ви відчуваєте себе трохи втраченим у джунглях, я пропоную перевірити React[13].

3.5.2. Простота

React.js просто простіше зрозуміти відразу. Компонентний підхід, чітко визначений життєвий цикл і використання простого JavaScript дозволяють реагувати дуже просто, навчитися, створювати професійні веб-сайти (і мобільні програми) і підтримувати його. React використовує спеціальний синтаксис під назвою JSX, який дозволяє змішувати HTML з JavaScript[14]. Це не є обов’язковим; Розробник все ще може писати на звичайному JavaScript, але JSX набагато простіше у використанні.

3.5.3. Легко навчатись

Той, хто володіє базовими знаннями в області програмування, може легко зрозуміти React, в той час як Angular і Ember називаються “предметно-орієнтованою мовою”, маючи на увазі, що їх важко вивчати. Для React.js вам просто необхідні базові знання CSS і HTML.

3.5.4.Нативний підхід

React можна використовувати для створення мобільних додатків (React Native). А React — відданий прихильник можливості повторного використання, що означає, що підтримується велике повторне використання коду. Так що в той же час ми можемо зробити IOS, Android і веб-додаток.

3.5.5.Прив'язка даних

React використовує однобічну прив'язку даних, а архітектура додатки, звана Flux, контролює потік даних до компонентів через одну контрольну точку — диспетчер. Простіше налагоджувати автономні компоненти великих додатків ReactJS.

3.5.6.Продуктивність

React не пропонує ніякої концепції вбудованого контейнера для залежності. Ви можете використовувати модулі Browserify, Require JS, EcmaScript 6, які ми можемо використовувати через Babel, ReactJS-di, щоб автоматично вводити залежності[15].

3.5.7.Тестуваність

Програми React.js дуже легко перевірити. Реактивні види можуть розглядатися як функції держави, тому ми можемо маніпулювати станом, ми переходимо до перегляду React.js і дивимося на вивід і спрацьовує дії, події, функції.

3.6.Віртуальний дом

Віртуальний DOM — це концепція програмування, де ідеальне або "віртуальне" подання UI зберігається в пам'яті і синхронізується з "реальним" DOM бібліотекою, такою як ReactDOM. Цей процес називається примиренням.

Віртуальний DOM — це відображення в пам'яті Real DOM. Це легкий об'єкт JavaScript, який є копією Real DOM[16].

ReactJS не оновлює Real DOM безпосередньо, а оновлює Virtual DOM. Це спричиняє велику користь для ReactJS.

3.6.1.Проблема

Маніпуляція DOM — це серце сучасної, інтерактивної мережі. На жаль, це також набагато повільніше, ніж більшість операцій JavaScript.

Ця повільність ускладнюється тим фактом, що більшість фреймворків JavaScript оновлюють DOM набагато більше, ніж вони повинні.

Наприклад, припустимо, що у вас є список з десяти елементів. Ви відмічаєте перший елемент. Більшість фреймворків JavaScript відновить весь список. Це в десять разів більше, ніж потрібно! Змінилося лише один елемент, але решта дев’ять перебудовуються саме так, як вони були раніше.

Відновлення списку не має великого значення для веб-браузера, але сучасні веб-сайти можуть використовувати величезну кількість маніпуляцій DOM. Неефективне оновлення стало серйозною проблемою.

Щоб вирішити цю проблему, люди в React популяризували те, що називається віртуальним DOM.

3.6.2.Вирішення проблеми

У React, для кожного об’єкта DOM, існує відповідний “віртуальний об’єкт DOM”. Віртуальний об’єкт DOM є поданням об’єкта DOM, як легка копія.

Віртуальний об’єкт DOM має ті ж властивості, що й реальний об’єкт DOM, але йому не вистачає реальної можливості безпосередньо змінювати те, що знаходиться на екрані.

Маніпулювання DOM відбувається повільно. Маніпулювання віртуальним DOM набагато швидше, тому що на екрані нічого не відбувається. Подумайте про маніпулювання віртуальним DOM як редагування плану, на відміну від переміщення кімнат у фактичному будинку.

3.6.3.Як це допомагає

Коли ви представляєте елемент JSX, кожен окремий об’єкт віртуального DOM оновлюється.

Це звучить неймовірно неефективно, але вартість незначна, тому що віртуальний DOM може оновлюватися так швидко.

Після оновлення віртуального DOM, React порівнює віртуальний DOM із знімком віртуального DOM, зробленим безпосередньо перед оновленням.

Порівнюючи нову віртуальну DOM з попередньою версією, React з’ясовує, які саме віртуальні об’єкти DOM змінилися. Цей процес називається “розбіжним”.

Як тільки React знає, які об’єкти віртуального DOM змінилися, то React оновлює ці об’єкти, і тільки ті об’єкти, на реальному DOM. У нашому прикладі з React було б достатньо розумним, щоб відновити ваш один відредагований список-елемент і залишити решту списку поодинці.

Це має велике значення! Реакція може оновлювати тільки необхідні частини DOM. Репутація компанії React за ефективність значною мірою пов’язана з цією інновацією.

Загалом, ось що відбувається, коли ви намагаєтеся оновити DOM у React:

— весь віртуальний DOM оновлюється;

— віртуальний DOM порівнюється з тим, як він виглядав, перш ніж оновити його. Реагуйте, які об’єкти змінилися;

— змінені об’єкти і тільки змінені об’єкти оновлюються на реальному DOM;

— зміни на реальному DOM призводять до зміни екрана.

3.7. Single Page Application

У минулому, коли браузери були набагато менш здатними, ніж сьогодні, а продуктивність JavaScript була слабкою, кожна сторінка надходила з сервера[17]. Кожен раз, коли ви натискали щось, новий запит був зроблений на сервер, і браузер згодом завантажив нову сторінку.

Тільки дуже інноваційні продукти працювали по-різному і експериментували з новими підходами.

Сьогодні, популяризуючи сучасні фреймворки JavaScript, такі як React, програма зазвичай створюється як додаток для однієї сторінки: ви завантажуєте код програми (HTML, CSS, JavaScript) лише один раз, і коли ви взаємодієте з програмою, що зазвичай відбувається JavaScript перехоплює події веб-переглядача і замість того, щоб робити новий запит на сервер, який потім повертає новий документ, клієнт запитує деякий JSON або виконує дію на сервері, але сторінка, яку бачить користувач, ніколи не буде повністю знищена, і веде себе більше як настільний додаток.

Програми для однієї сторінки вбудовані в JavaScript (або, принаймні, зібрані в JavaScript) і працюють у браузері.

Технологія завжди однакова, але філософія і деякі ключові компоненти того, як працює програма, різні.

3.7.1.Переваги і недоліки SPA

Користувач SPA відчуває себе набагато швидше, тому що замість того, щоб очікувати, що відбудеться зв’язок клієнт-сервер і чекати, поки веб-переглядач знову відобразить сторінку, тепер можна отримати миттєвий відгук. Це відповідальність виробника програм, але ви можете мати переходи і пряди і будь-яке поліпшення UX, що, безумовно, краще, ніж традиційний робочий процес.

На додаток до швидшого використання досвіду користувачеві, сервер споживатиме менше ресурсів, оскільки ви можете зосередитися на забезпеченні ефективного API, замість того, щоб створювати серверні частини макетів.

Це робить його ідеальним, якщо ви також створюєте мобільну програму поверх API, оскільки ви можете повністю повторно використати існуючий серверний код.

Програми для однієї сторінки легко перетворюються на Прогресивні веб-програми, що, в свою чергу, дає змогу кешувати локально та підтримувати автономний режим для ваших служб (або краще повідомлення про помилку, якщо користувачі мають бути онлайн).

SPAs найкраще використовувати, коли немає необхідності SEO (search engine optimization). Наприклад, для програм, які працюють за логіном[18].

Пошукові системи, вдосконалюючись щодня, все ще мають проблеми з індексацією сайтів, побудованими з підходом SPA, а не традиційними сторінками, що надаються сервером. Це стосується блогів. Якщо ви збираєтеся покладатися на пошукові системи, не варто навіть створювати додаток для однієї сторінки, не маючи також частини, що надає сервер.

При кодуванні SPA ви пишете багато JavaScript. Оскільки програма може бути довготривалою, вам потрібно буде приділяти більше уваги можливим витокам пам’яті — якщо в минулому ваша сторінка мала час життя, що враховувалася в хвилинах, тепер SPA може залишатися відкритим протягом декількох годин час і якщо є які-небудь проблеми з пам’яттю, що збільшить використання пам’яті браузера набагато більше, і це призведе до неприємного повільного досвіду, якщо ви не піклуєтесь про нього.

SPA-центри прекрасні при роботі в команді. Розробники Backend можуть зосередитися лише на API, а розробники інтерфейсу можуть зосередитися на створенні найкращого користувальницького досвіду, використовуючи вбудований в бекенд API.

В якості однієї сторінки, програми з однією сторінкою значною мірою покладаються на JavaScript. Це може призвести до поганого досвіду використання програми, що працює на пристроях з низьким енергоспоживанням. Крім того, деякі відвідувачі можуть вимкнути JavaScript, а також потрібно розглянути доступність для всього, що ви збираєте.

3.7.2.Приклади застосування Single Page Applications

Деякі помітні приклади:

— Gmail;

— Google Maps;

— Facebook;

— Twitter;

— Google Drive.

3.7.3.Перевизначення навігації

Оскільки ви позбавляєтеся від навігації браузера за умовчанням, URL-адреси потрібно керувати вручну.

Ця частина програми називається маршрутизатором. Деякі фреймворки вже піклуються про них (як Ember), інші вимагають бібліотек, які будуть виконувати цю роботу (наприклад, React Router).

В чому проблема? Спочатку це було запізненням для розробників, які створювали програми з однієї сторінки. Це призвело до виникнення загальної проблеми з “кнопкою зворотного ходу”: при навігації всередині програми URL не змінювався (так як навігація за замовчуванням браузера була захоплена) і натиснувши кнопку “назад”. може перейти на веб-сайт, який ви відвідали давно.

Цю проблему тепер можна вирішити за допомогою API-інтерфейсу History, який пропонують веб-переглядачі, але більшу частину часу ви будете використовувати бібліотеку, яка внутрішньо використовує цей API, наприклад React Router.

3.8.Бібліотека реалізації керування станом системи Redux

Redux — не що інше, як сховище, яке містить стан програми. Це стає болісним завданням, коли розмір програми стає великим, щоб керувати станом кожного компонента вашої програми. Отже, редукція приходить на допомогу, підтримуючи та оновлюючи стан кожного компонента нашої програми[19].

Redux часто буває незрозумілим, коли ми вперше пробуємо свої сили. Отже, я наведу приклад, щоб ви зрозуміли, що таке redux і чому нам взагалі потрібно редукції.

У реактивному додатку все є компонентом. Уявіть, як важко стає, якщо у вашому додатку дуже багато компонентів, подібних до наведених на рисунку 3.3, тому стає важко керувати потоком даних від батьківських до дочірніх компонентів.

Це перша причина, чому ми використовуємо redux, оскільки вона керує станами всіх компонентів для нас.

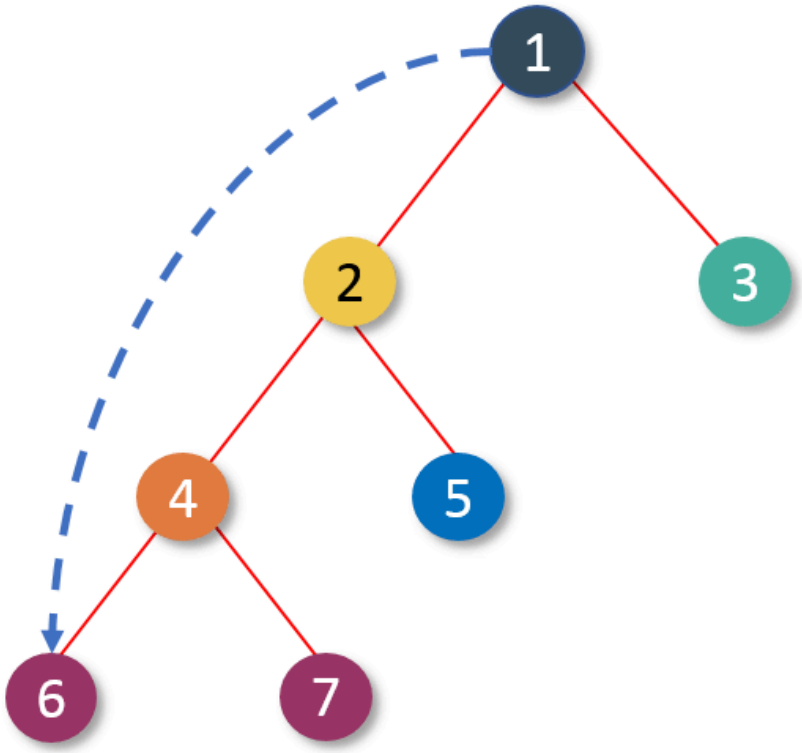

Рисунок 3.3 — Зразок дерева компонентів React

У реактивному застосуванні існує односпрямований потік даних за односпрямованим я маю на увазі потоки даних від батьківських компонентів до дочірніх компонентів не навпаки, так що ви посилаєте дані з батьківських компонентів до дочірніх компонентів у вигляді того, що ми називаємо реквізитами, тоді цей дочірній компонент використання цієї підставки.

Redux — це бібліотека відкритого коду JavaScript для керування станом програми.

Після виконання дії в цьому випадку замовлення взуття є акцією, яку ви чекаєте на доставку, але чи відбувається це так, як тільки ви замовляєте щось із фліпкарту, ви отримаєте доставку відразу. Ні, насправді це вимагає часу, і є процес, який слідує кожен раз, коли ви замовляєте щось з вашого улюбленого веб-сайту.

Таким чином, подібно до редуксу після виконання дії, існує термін, що називається диспетчером, який посилає вашу дію на редуктор. Так само, як і після

розміщення замовлення, ваш пакет буде відправлений у найближчий склад на вашу адресу. Така ж робота в редукції здійснюється шляхом відправки.

Існує декілька понять, які потрібно розуміти для розуміння редуксу. Я постараюся пояснити їх за допомогою прикладу (рисунок 3.4). Припустимо, ви замовили пару взуття з фліпкарту після замовлення взуття, яку ви отримуєте від агента доставки в певний час. Отже, ваше замовлення взуття — це дія, яка є однією з концепцій редуксу.

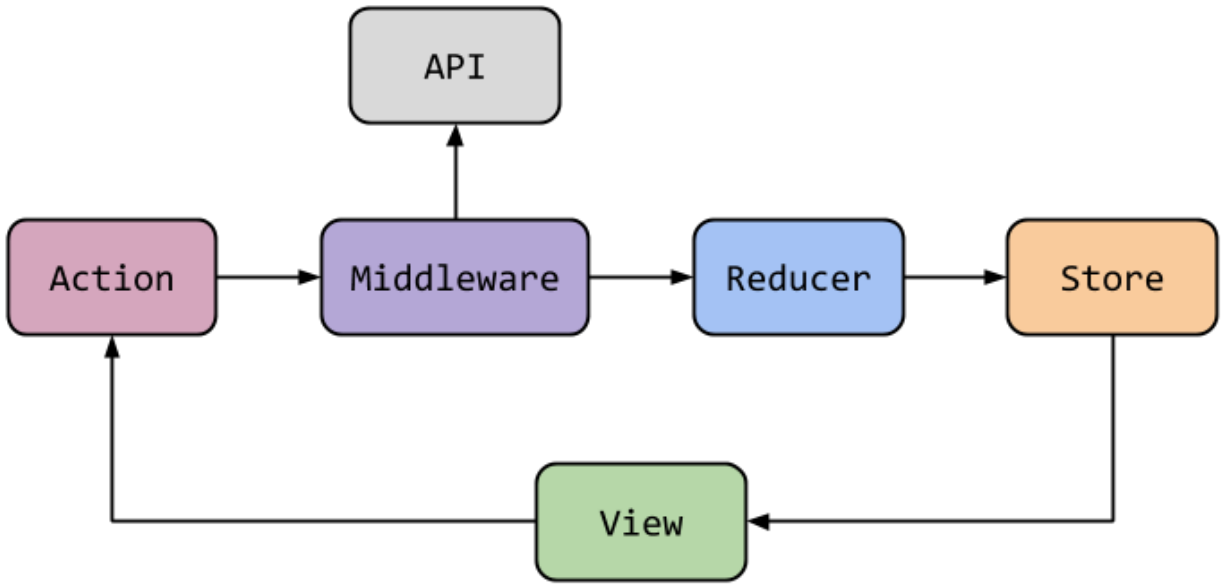

Рисунок 3.4 — Архітектура Redux

Тепер Reducer дивиться на дію і відповідно робить те, що потрібно для зберігання даних. Редуктор — це не що інше, як файл, що складається з оператора case case і використовується для зберігання даних у сховищі і повернення оновленого значення стану з магазину. Тому, коли стан оновлюється, значення в магазині теж оновлюється.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Даний програмний продукт був розробений та протестований у програмному середовищі розробки WebStorm. Також дана система має клієнт-серверну архітектуру (рисунок 4.1), яка складається з сервера, веб-клієнта та бази даних.

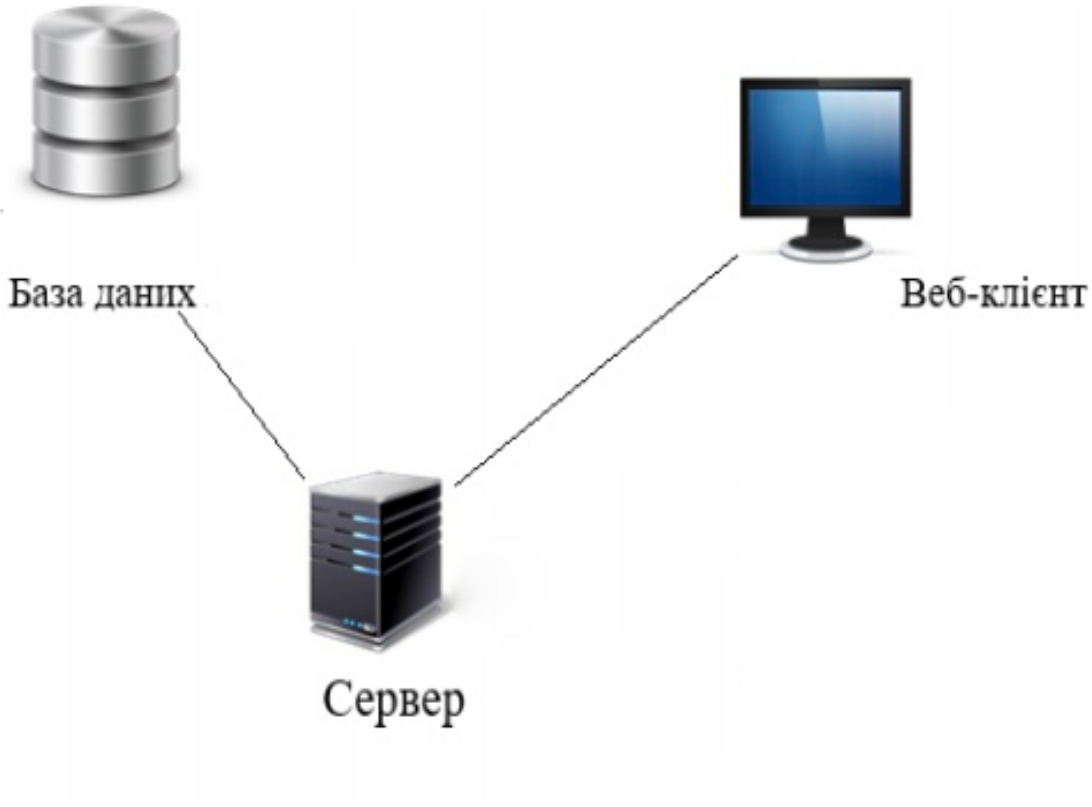

Рисунок 4.1 — Архітектура системи

Дана система складається з таких частин:

— веб-клієнт React.js;

— сервер Express.js;

— база даних MySQL.

4.1.Опис функціональності системи

Модуть “Студент” веб-застосунку системи управління дипломними проектами має одного актора, яким є студент. На рисунку 4.2 представлена діаграма прецедентів, яка описує дії і функцію студента у системі.

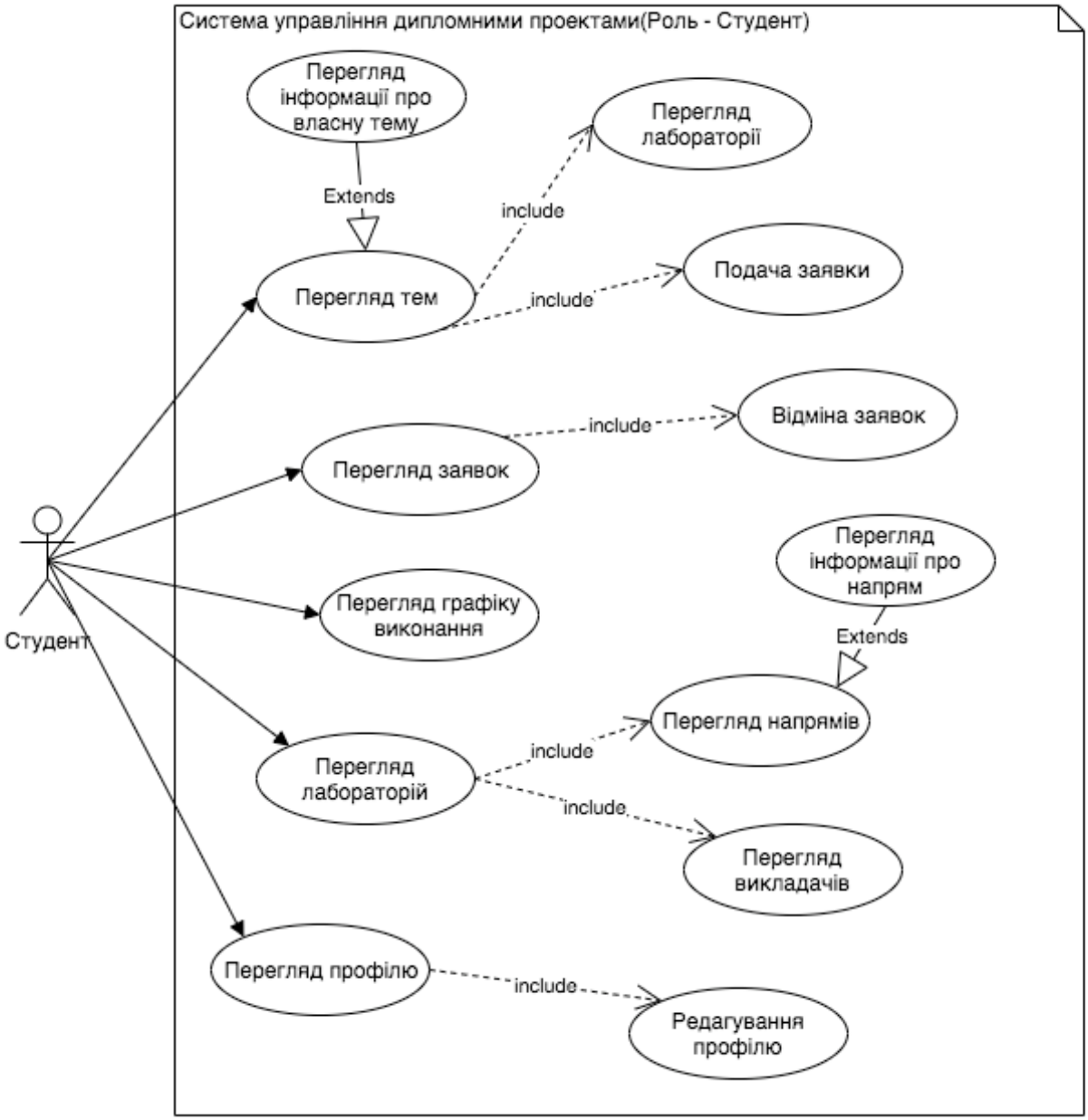

Рисунок 4.2 — Діаграма прецедентів веб-застосунку

Користувачеві надається такі функції:

— перегляд профілю;

— редагування профілю;

— перегляд лабораторій;

— перегляд викладачів;

— перегляд напрямків лабораторії;

— перегляд інформації про напрями лабораторії;

— перегляд графіку виконання;

— перегляд заявок;

— відміна заявок;

— подача заявок;

— перегляд тем дипломних робіт;

— перегляд інформації про власну тему;

— авторизація в систему.

4.2.База даних

База даних побудова на СКБД MySQL[21]. Архітектура бази повністю відповідає тематиці роботи. На рисунку 4.3 представлена схема бази даних:

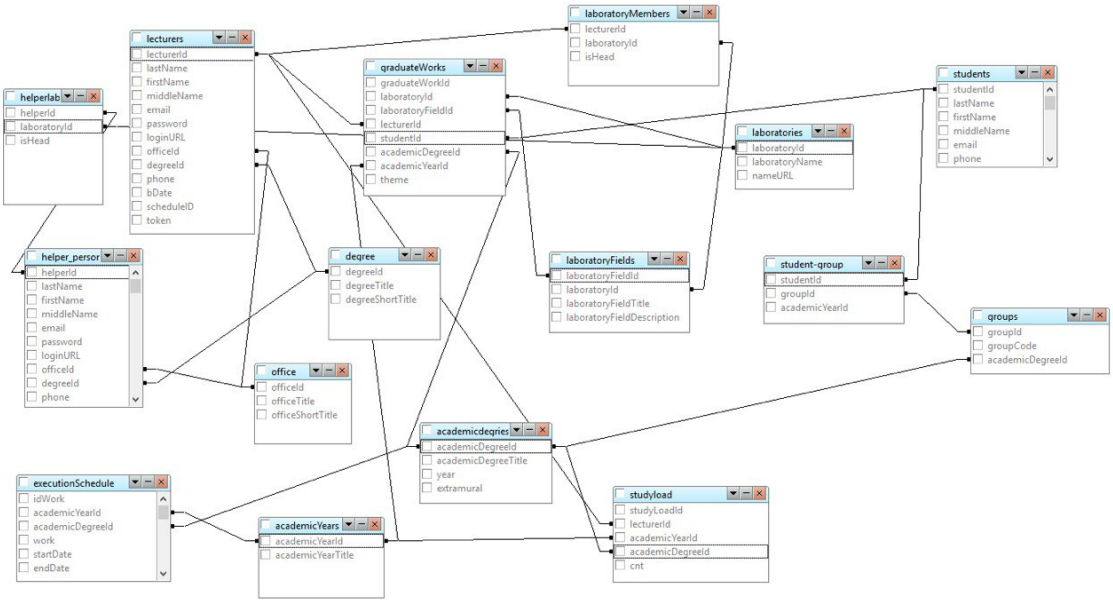

Рисунок 4.3 — Схема бази даних

4.3.Створення веб-застосунку за допомогою React.js

Для забезпечення зручності роботи з даними було розроблено веб-застосунок, написаний за допомогою мови програмування JavaScript. Це розв’язує проблему кросплатформності та дозволяє запустити програму на будь-якому пристрої, що має вихід в інтернет.

Для створення проекту потрібно встановити бібліотеку create-react-app. Вона полегшує створення коду додатків і бібліотеки, а також виконання різноманітних поточних завдань розробки, таких як тестування, комплектація та розгортання.

Для створення нового проекту потрібно виконати команду create-react-app project_name і тоді створиться новий проект в поточній директорії з всіма необхідними залежностями і також сервером, так що ви можете легко створювати і обслуговувати вашу програму локально. Команда “yarn start” запускає сервер, стежить за файлами і перебудовує додаток, коли ви зробите зміни в цих файлах. Параметр --open(або просто -o) автоматично відкриває ваш браузер за адресою http://localhost:3000/.

Основною фундаментальною одиницею React є компоненти, які використовуються в якості будівельних блоків. Компоненти можуть приймати різні параметри. Ці параметри називаються props.

Назви стилів і значення, як правило, відповідають тому, як CSS працює у веб додатках, крім способу запису назви — в React використовується camelCase стиль, наприклад backgroundColor, а не background-color[20]. Те ж саме стосуються назв подій.

В папці components знаходяться основні компоненти застосунку, в папці configs — налаштування, components — сторінки застосунку, routes — маршрутизація, store — управляння станом системи, styles — основні стилі, utils — допоміжні функції.

Для управління станом системи було використано бібліотеку Redux. Redux — це контейнер станів для застосунків JavaScript. Він допомагає розробникам оптимізувати код програми. Крім того, він забезпечує вдосконалення досвіду

розробника, наприклад, редагування живого коду в поєднанні з відладчиком, що працює під час роботи.

На рисунку 4.4 представлена файлова структура веб-клієнта:

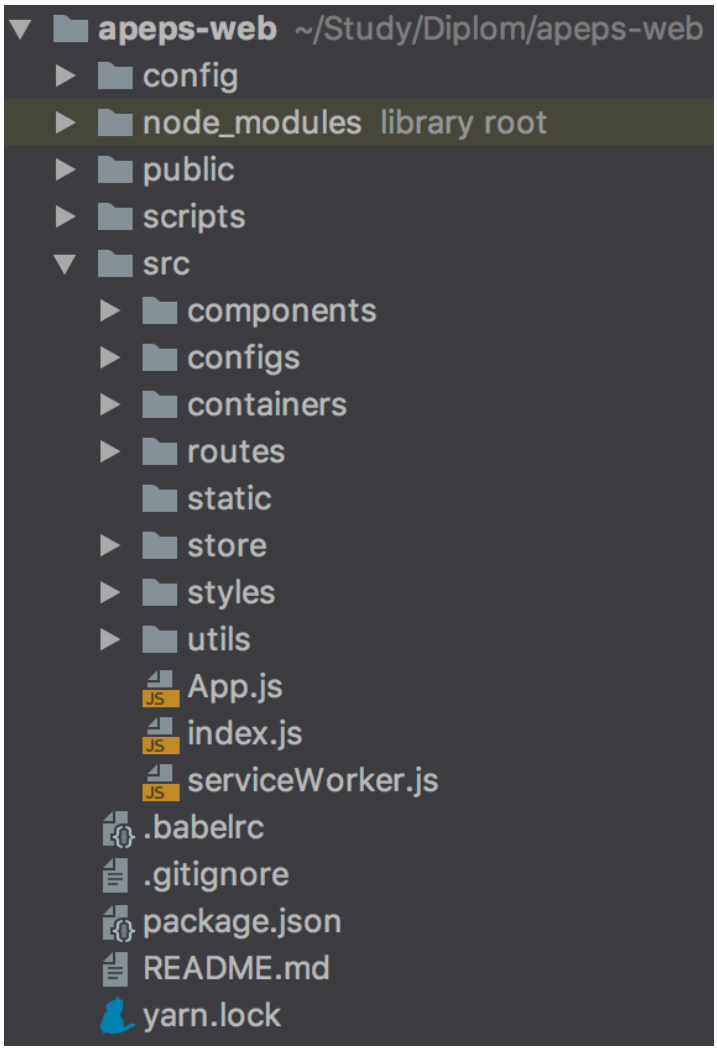

Рисунок 4.4 — Файлова структура веб-клієнта

Було сконфігуровано store за допомогою функції createStore бібліотеки redux, в яку прокидуються параметрами reducer і enhancer, де reducer це rootReducer, а в ролі enhancer виступає функція composeWithDevTools бібліотеки redux-devtools-extension для того щоб можна було в браузері відслідковувати зміни store і яка приймає в якості параметрів middlewares за допомогою функції applyMiddleware з бібліотеки redux додаємо sagaMiddleware, а також createLogger з бібіліотеки redux-logger, якщо ми знаходимиось в режині розробки. При ініціалізації store запускається sagaMiddleware.

Store (рисунок 4.5) складається з actions, які надсилають дані з програми в store, є також єдиним джерелом інформації для store і для того, щоб їх відправити необхідно виконати команду store.dispatch().

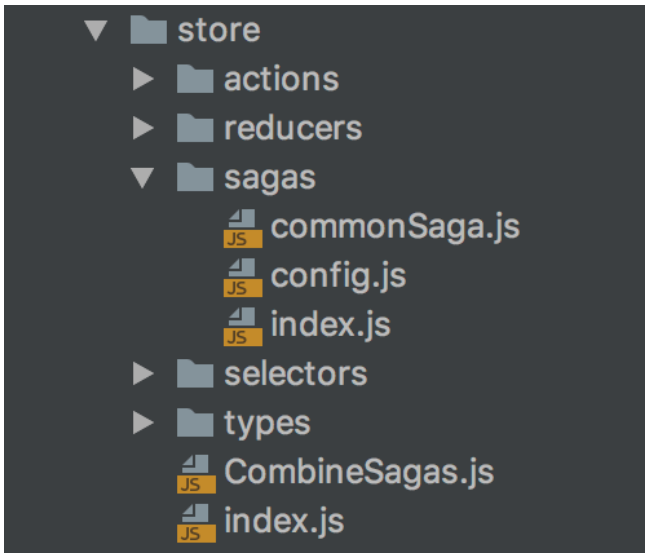

Рисунок 4.5 — Файлова структура системи керування станом системи

Також Store містить reducers, які вказують як змінюється стан програми у відповідь на actions надіслані до store. Корисно пам’ятати, що дії не описують, як змінюється стан програми, а лише описують те, що сталося. Reducers є чистими функціями — детермінована функція без побічних ефектів. Детермінована функція — функція, яка завжди повертає однакове значення при певному вводі. Побічна функція — функція, яка змінює щось за границями функції. Потім всі reducers збирається в один — rootReducer за допомогою функції combineReducers бібліотеки redux.

Такоє є типи actions, які потрапляють в reducers і їх виносять в змінні для кожного модуля.

Selectors містить селектори строрені для мемоізації даних. Reselect має функцію createSelector для створення мемоізованих селекторів. Селектор перераховувати дані тільки тоді коли один з його аргументів змінився. Селектори можуть використовуватись в якості параметрів для інших селесторів.

В файлі. config.js прописані налаштування для запитів на сервер, а в файлі commonSaga.js написана універсальна saga, яка яка приймає на всіх всі actions, які

містять слово _REQUEST кидає запит на сервер, обробляє його в залежності від налаштувань прописаних в файлі config.js.

Для взаємодії store з компонентами react використовується функція вищого порядку connect з бібліотеки react-redux. Функція вищого порядку — це функція, що повертає іншу функцію і можу приймати як аргументи інші функції. Функція connect приймає в якості параметрів mapStateToProps і mapDispatchToProps.

Для того, щоб react знав про store потрібно обернути наш застосунок в Provider з бібліотеки react-redux, який приймає в якості props store.

В якості маршрутизатора було використано компонент Router з бібліотеки React Router. Компонент Router очікую тільки один елемент в якості дочірнього. Router приймає в якості props history, яка дозволяє з легкістю керувати історією сесії застосунку. Створено конпонент App який рендерить весь застосунок.

Для написання стилів було використано бібліотеку styled-components, яка також дозволяє використання різних тем зі стилями. Перевагою даного підходу є те, що не потрібно створювати css файли.

Для сповіщень було вирішено написати компонент NotificationProvider, який використовує React Context API.

Було написано компонент модального вікна за допомогою Portals, які відтворюють вузол дом поза ієрархію дом батьківського компонента.

4.4.Створення серверної частини за допомогою Express.js

Сервер було побудовано за допомогою платформи Node.js. Ця платформа дозволяє за допомогою мови програмування Javascript будувати rest api сервіси[22].

Для створення проекту потрібно встановити express, створити екземпляр застосунку і вказати порт на якому буде запущений застосунок.

Для обробки http post запиту в Express.js версії 4 і вище вам необхідно встановити модуль проміжного програмного забезпечення, званий body-parser[23].

Модуль body-parser витягує всю частину тіла вхідного потоку запитів і виставляє його на req.body.

Проміжне програмне забезпечення раніше входило до складу Express.js, але тепер його потрібно встановлювати окремо.

Цей модуль body-parser аналізує дані в кодуванні JSON, буфера, рядки і URL, відправлені з використанням http post запиту.

Для можливості робити запити на сервер з іншого домену потрібно встановити модуль cors.

Для перехвату помилок потрібно написати middleware(функцію проміжної обробки), яка перехоплює помилки і в результаті передає дану помилку.

Для надання статичних файлів використовується функція проміжної обробки express.static[24]. Для того, щоб дати можливість доступу до файлів потрібно вказати директорію в якій лежать статичні файли.

Для автоматичного перезапуску сервера коли змінивсся якийсь файл потрібно встановити модуть nodemon.

Для запуску сервера необхідно виконати команду nodemon app.js.

Сервер виконує такі функції:

— виконує зв’язок з базою даних

— виконує завантаження файлів;

— виконує відправку електронних листів;

— реалізує зв’язок з клієнтською частиною за допомогою http запитів.

В папці api знаходиться модуль маршрутизації і модулі системи. В папці db знаходиться модуль до бази даних. В папці mail знаходиться модуль відправки електронних листів. В папці public знаходяться статичні файли. В папці node_modules знаходяться встановлені залежності для розробки сервера.

Файл package.json містить список підключених модулів для даного проекту з npm, команди для запуску проекту. Список модулів з package.json:

— body-parser версії ^1.18.3;

— cors версії ^2.8.5;

— express версії ^4.16.4;

— multer версії ^1.4.1;

— mysql версії ^2.17.1;

— nodemailer версії ^6.1.1;

— nodemon версії ^1.19.0.

Файлову структуру сервера зображено на рисунку 4.6:

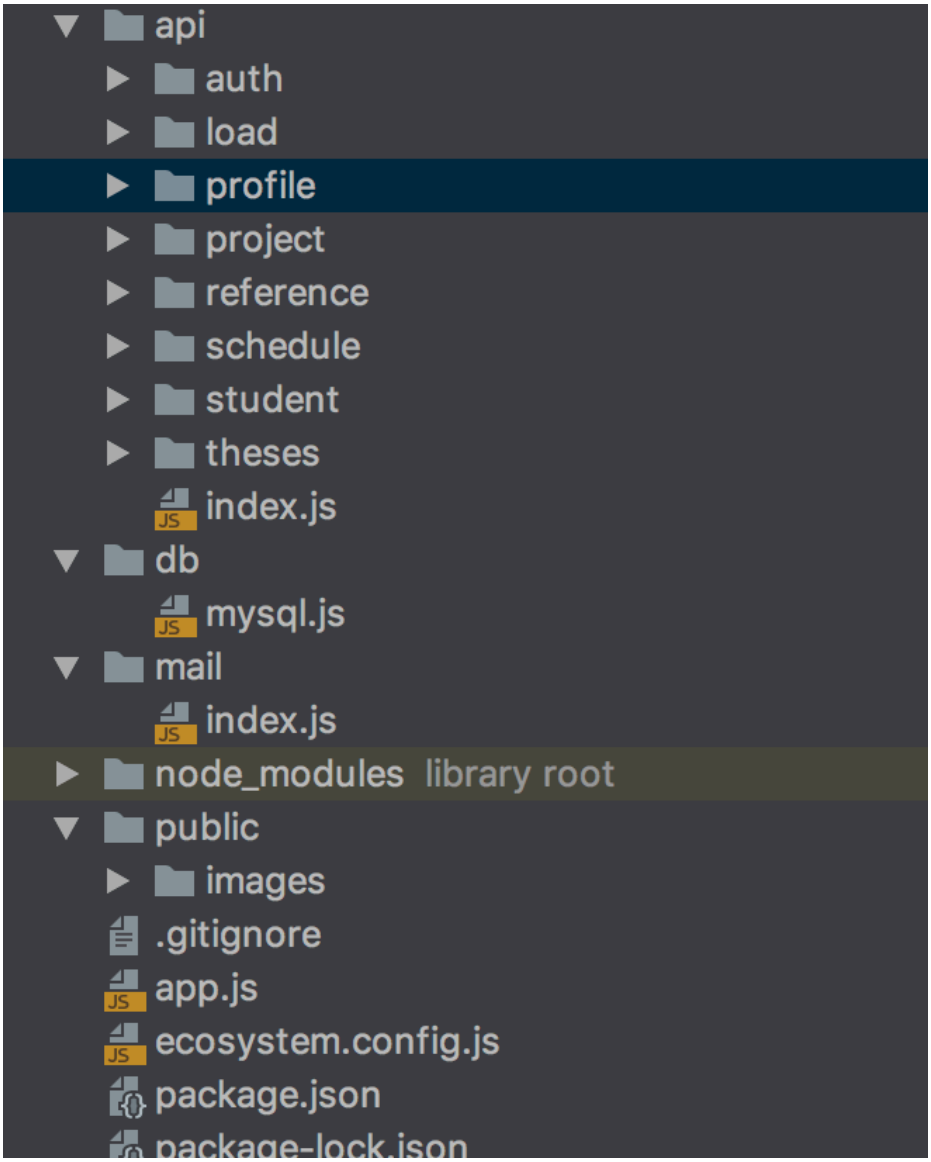

Рисунок 4.6 — Файлова структура сервера

Для того щоб додати функціональну можливість підключення бази даних MySQL до додатка Express, необхідно всього лише завантажити в ваш додаток модуль mysql.

Для завантаження файлів на сервер потрібно встановити модуль multer, встановити налаштування по типу і розміру файлів, вказати нову назву файла і директорію куди він буде зберігатись.

Найбільша відмінність, яке ви можете тут помітити, полягає в тому, що Express за умовчанням дає вам роутер. Вам не потрібно вручну розбирати URL, щоб

вирішити, що робити, замість цього ви визначаєте маршрутизацію програми за допомогою app.get, app.post, app.put і так далі, а вони вже транслюються в відповідні HTTP-запити[25].

Одна з найпотужніших концепцій, яку реалізує Express — це патерн Middleware.

Для опрацювання помилок в Express ви повинні створити спеціальний проміжний обробник — middleware з чотирма вхідними параметрами. Оброблювач помилок повинен бути останньою функцією, доданої за допомогою app.use і приймає коллбек next. Він може використовуватися для об’єднання декількох обробників помилок.

5. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для використання реалізованого веб-застосунку необхідно мати встановленним будь-який Інтернет браузер. Потрібно переконатися, що ваша система відповідає данним вимогам. Хоча це, як правило, не є проблемою, коли мова йде про вимоги до апаратних засобів, ви можете помітити, що це, наприклад, зовсім інша історія, коли мова йде про підтримувані операційні системи. Користувачі Firefox у Windows 2000, наприклад, помітять, що вони не зможуть оновитись з Firefox 12 на 13 у найближчому майбутньому, оскільки Mozilla знизила підтримку цієї операційної системи, починаючи з цієї версії браузера. Підтримка браузера Google Chrome операційними системами починається з Windows XP SP2, OS X 10.5.6, Ubuntu 10.04, Debian 6, OpenSuse 11.3 та Fedora Linux 14.

Щоб реалізувати доступ до системи лише авторизованим корисувачам, було розробленно авторизацію (рисунок 5.1).

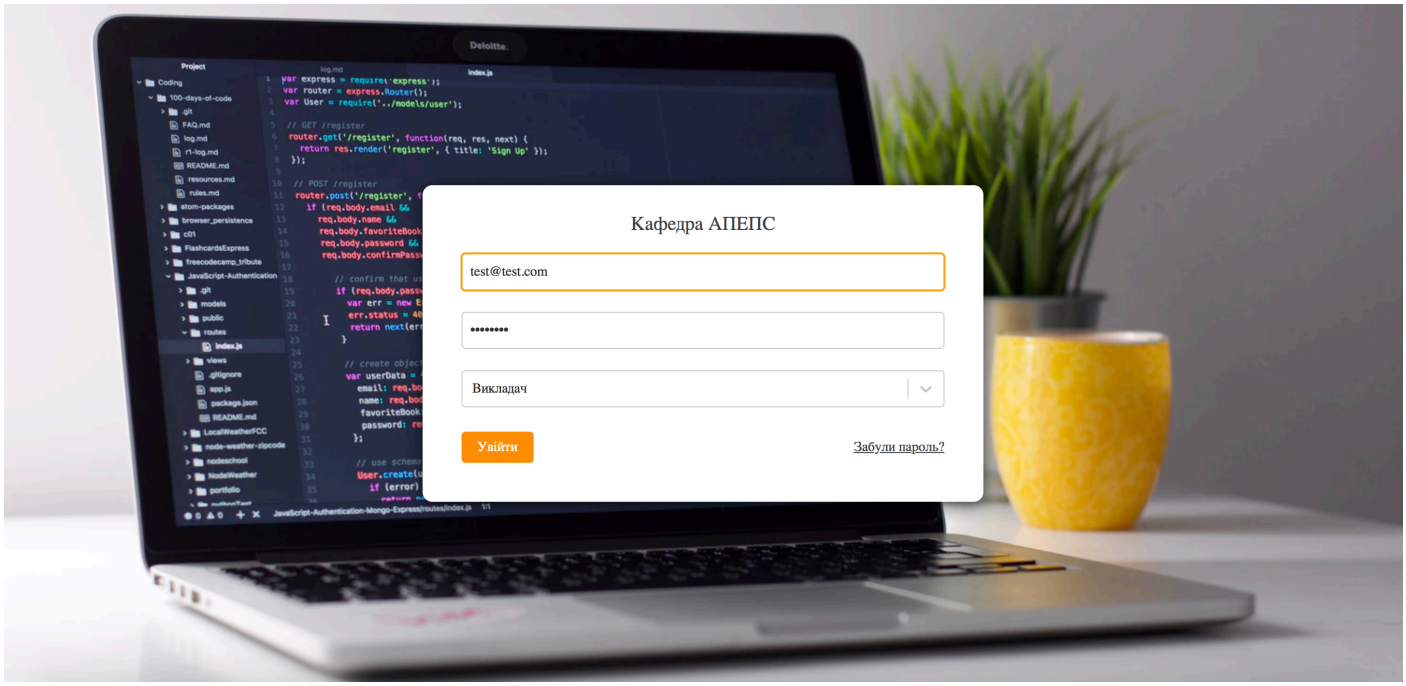

Рисунок 5.1 — Сторінка авторизації

Якщо користувач забув пароль, то він має можливість надіслати пароль на свою електронну адресу (рисунок 5.2).

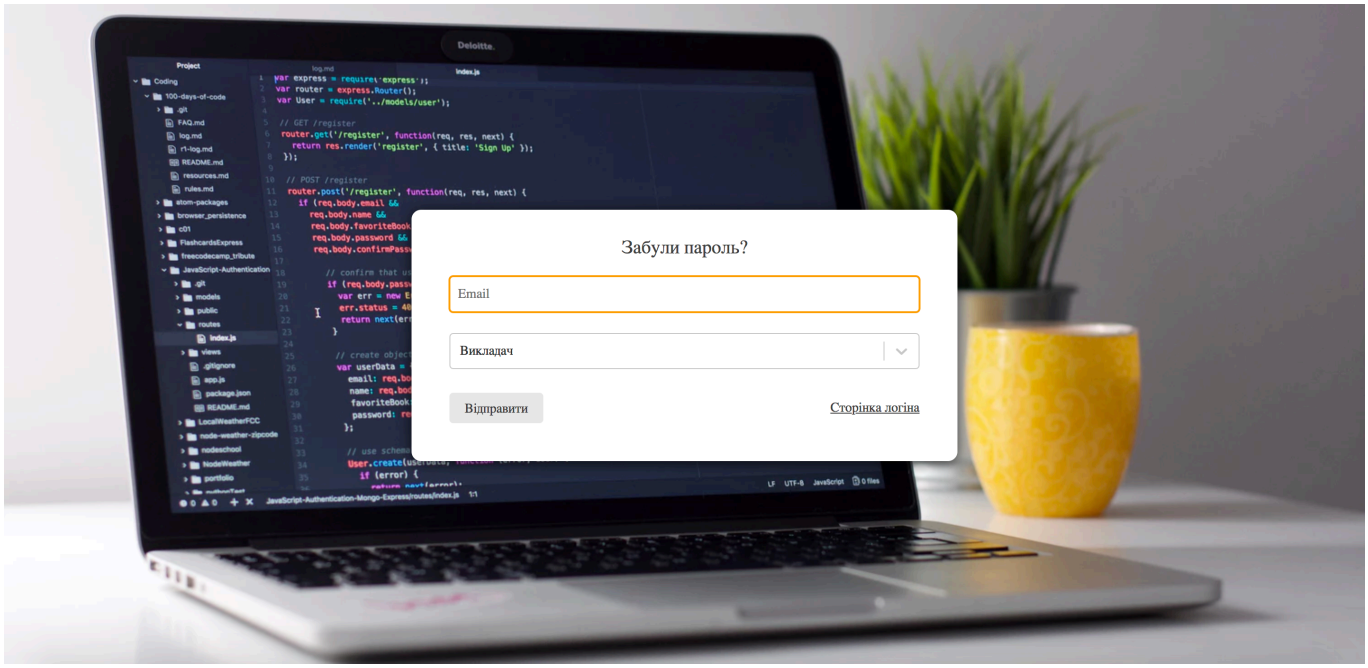

Рисунок 5.2 — Сторінка відновлення паролю

Після успішної авторизації корисувач має змогу шукати, переглядати дипломні роботи, відфільтровувати дані за лабораторією, піблабораторією і викладачем, а також виконувати пошук по дипломних роботах подавати заявки на декілька дипломних робіт (рисунок 5.3).

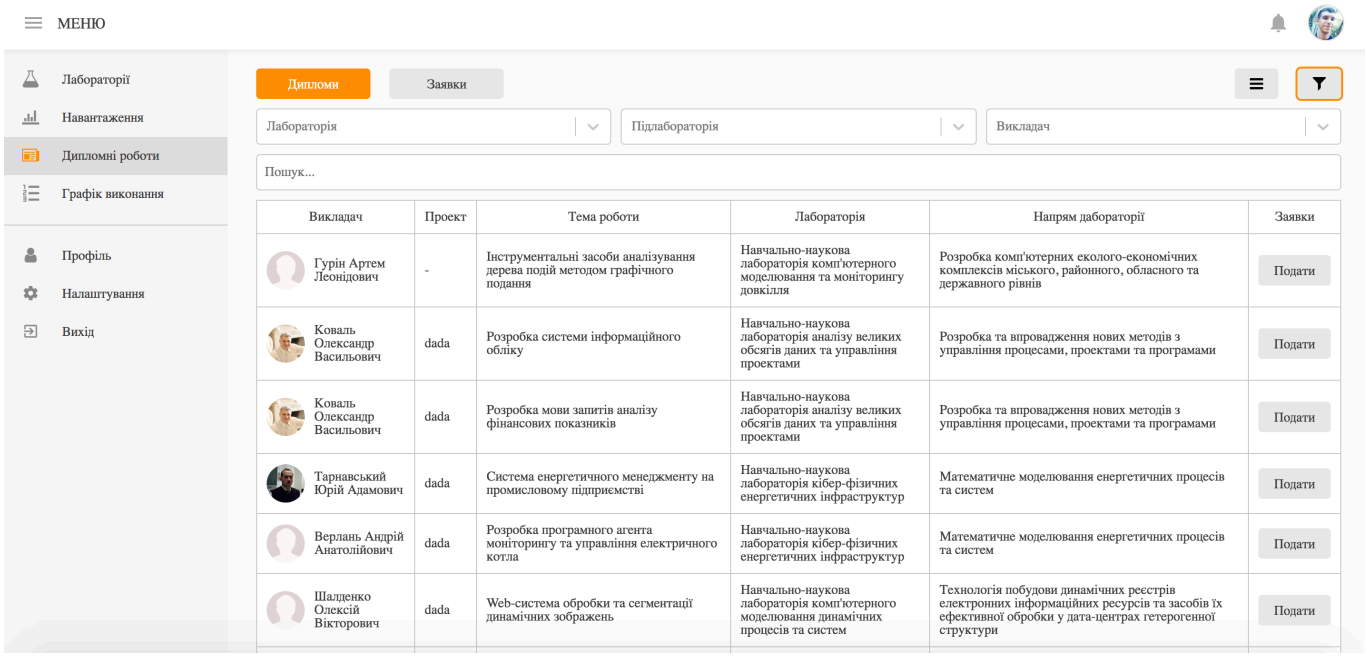

Рисунок 5.3 — Сторінка тем дипломних робіт

Студент має можливість подавати заявки максимум на 7 дипломних робіт. Для підтвердження надсилання заявки студент повинний підтвердити свої наміри відправити заявку в модальному вікні (рисунок 5.4).

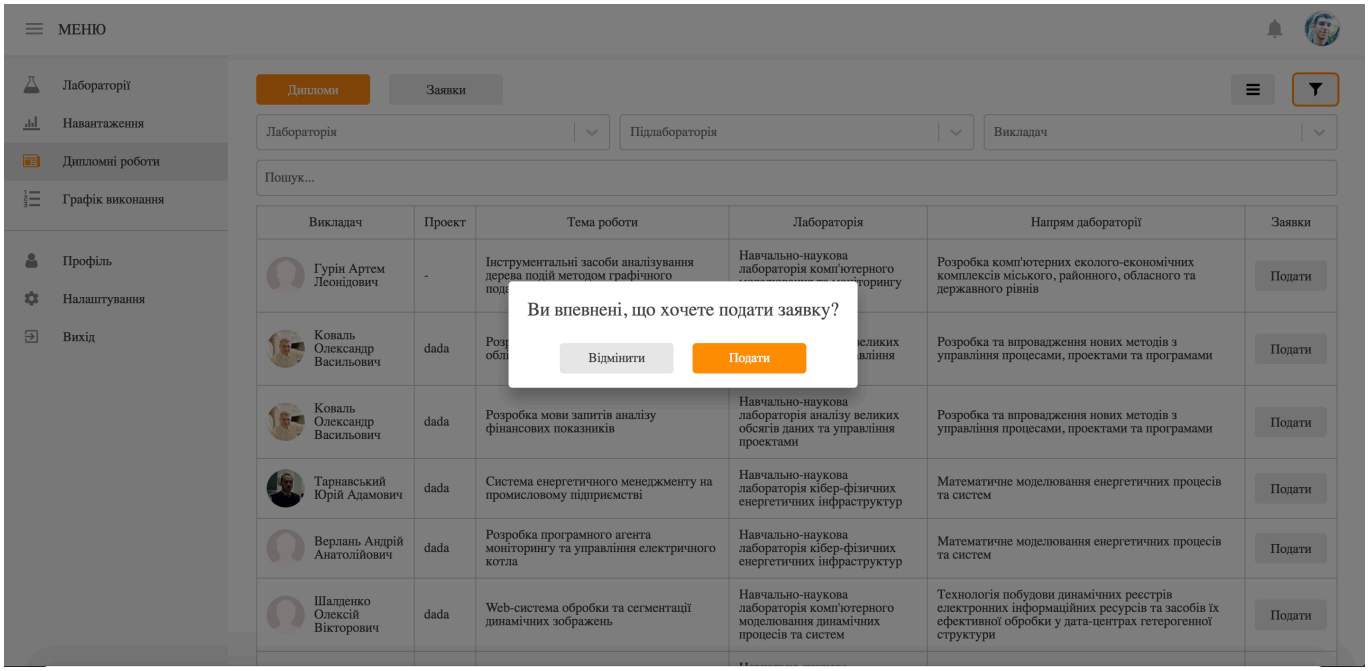

Рисунок 5.4 — Модальне вікно підтвердження відправки заявки

Студент має можливість переглядати свої заявки. Також є зміна вигляду заякок і дипломних робіт (рисунок 5.5).

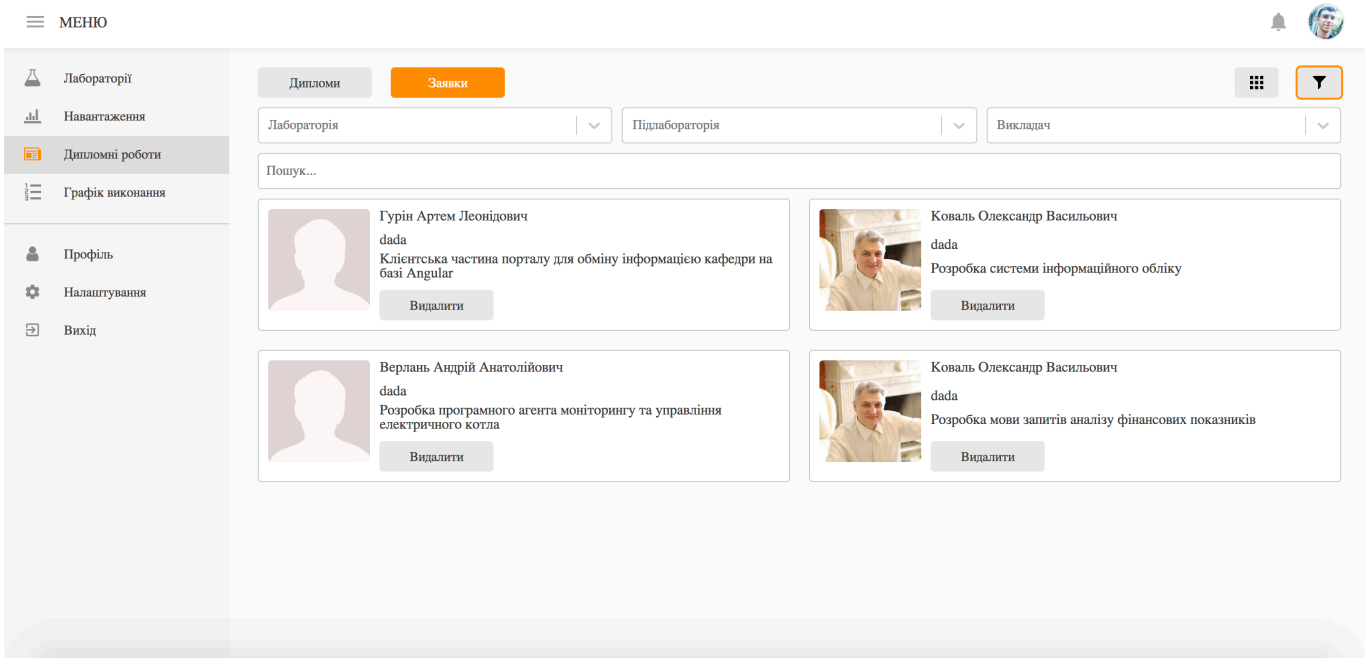

Рисунок 5.5 — Сторінка заявок на дипломні роботи

Для видалення заявки студент має підтвердити свої наміри на видалення. Застусок має крос-браузерну і адаптивну верстку, а тому його можна використовувати на мобільних пристроях (рисунок 5.6).

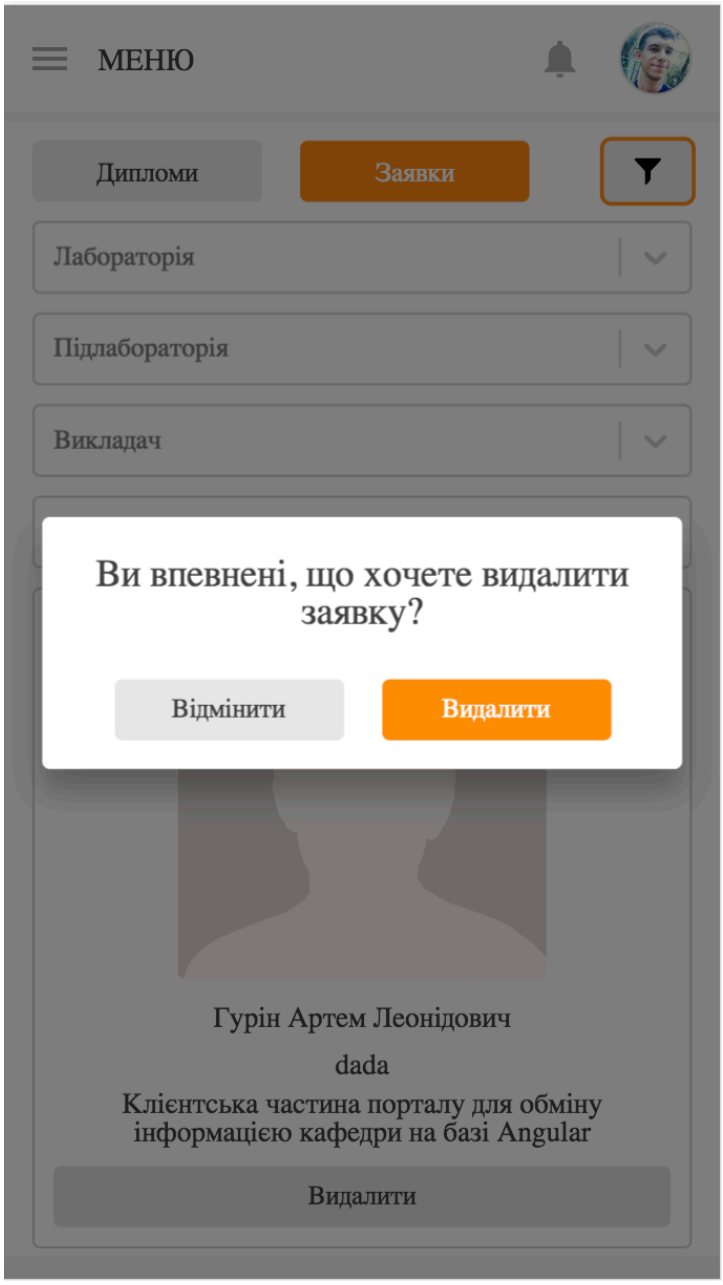

Рисунок 5.6 — Модальне вікно підтвердження видалення заявки

Після того як викладач прийняв заявку студента, студент має змогу переглядати інформацію про свою дипломну роботу (рисунок 5.7), а саме:

— контактні дані викладача;

— інформацію про тему дипломної роботи;

— графік виконання дипломної роботи.

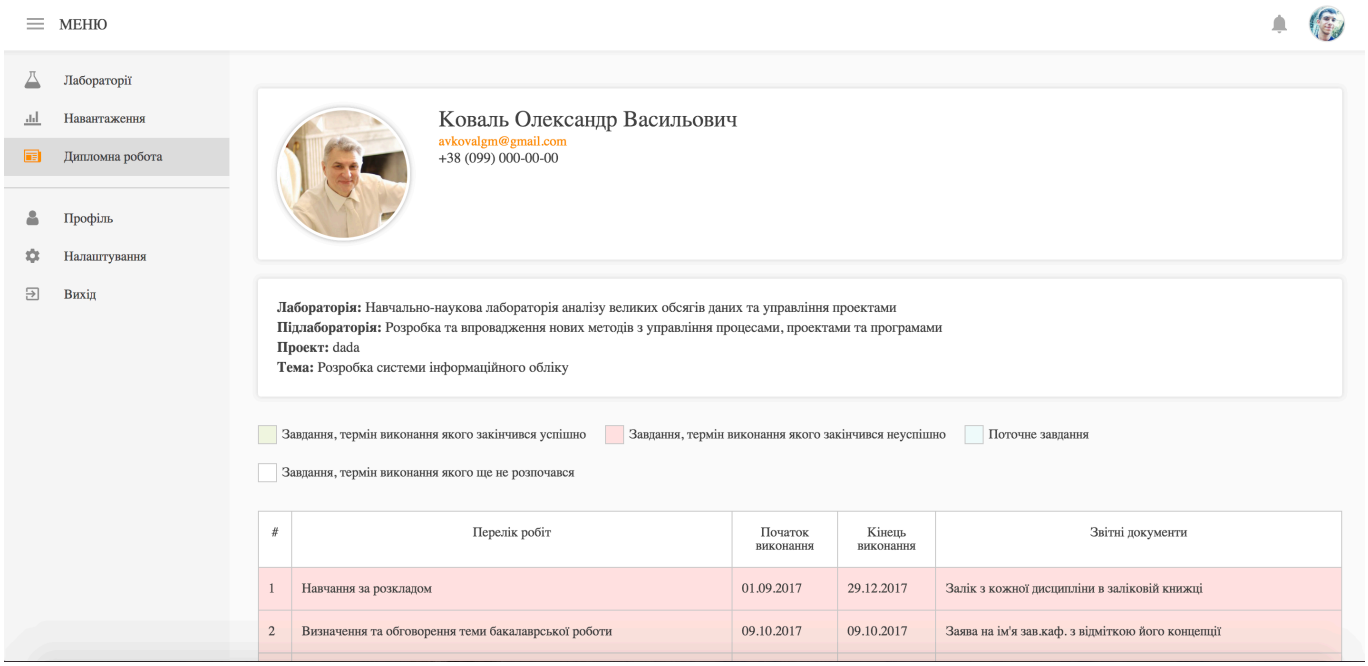

Рисунок 5.7 — Зразок інтерфейсу дипломної роботи студента

Студент має можливість переглядати свій профіль (рисунок 5.8).

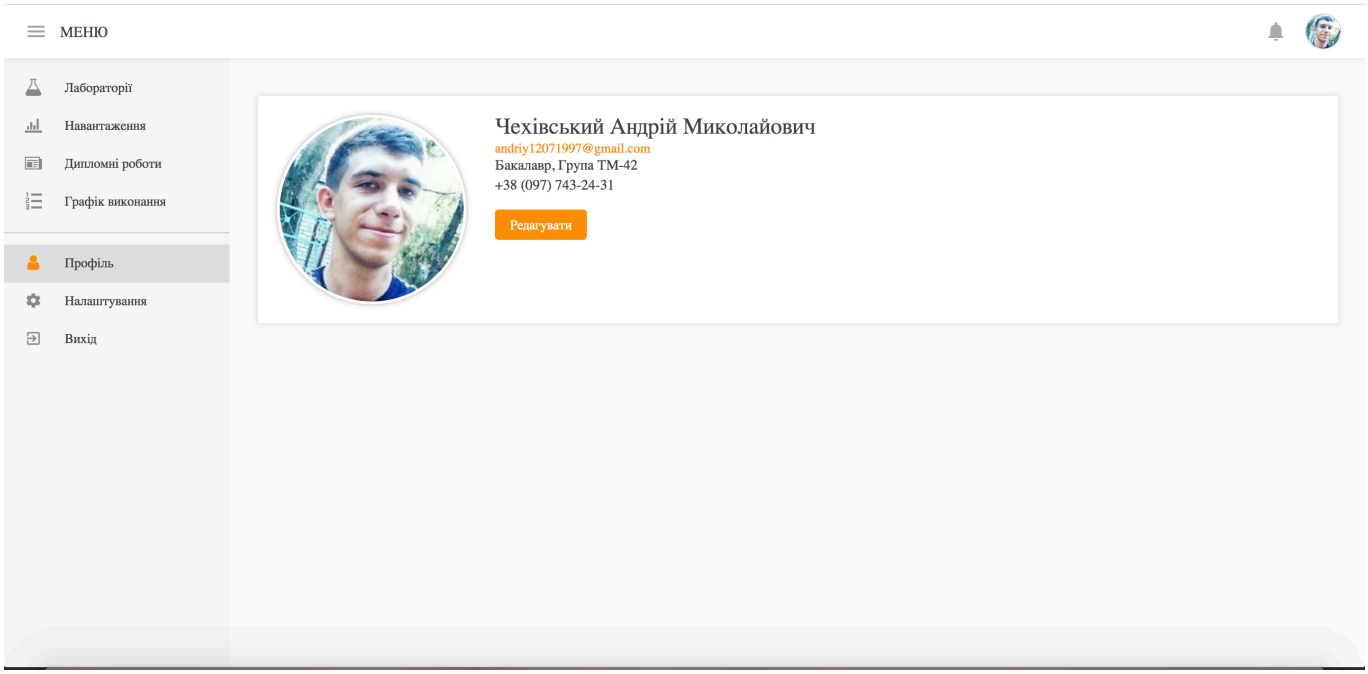

Рисунок 5.8 — Зразок інтерфейсу профіля студента

Студент має можливість змінювати свій аватар. При нажатті на аватар користувача пропонується вибрати фотографію (рисунок 5.9).

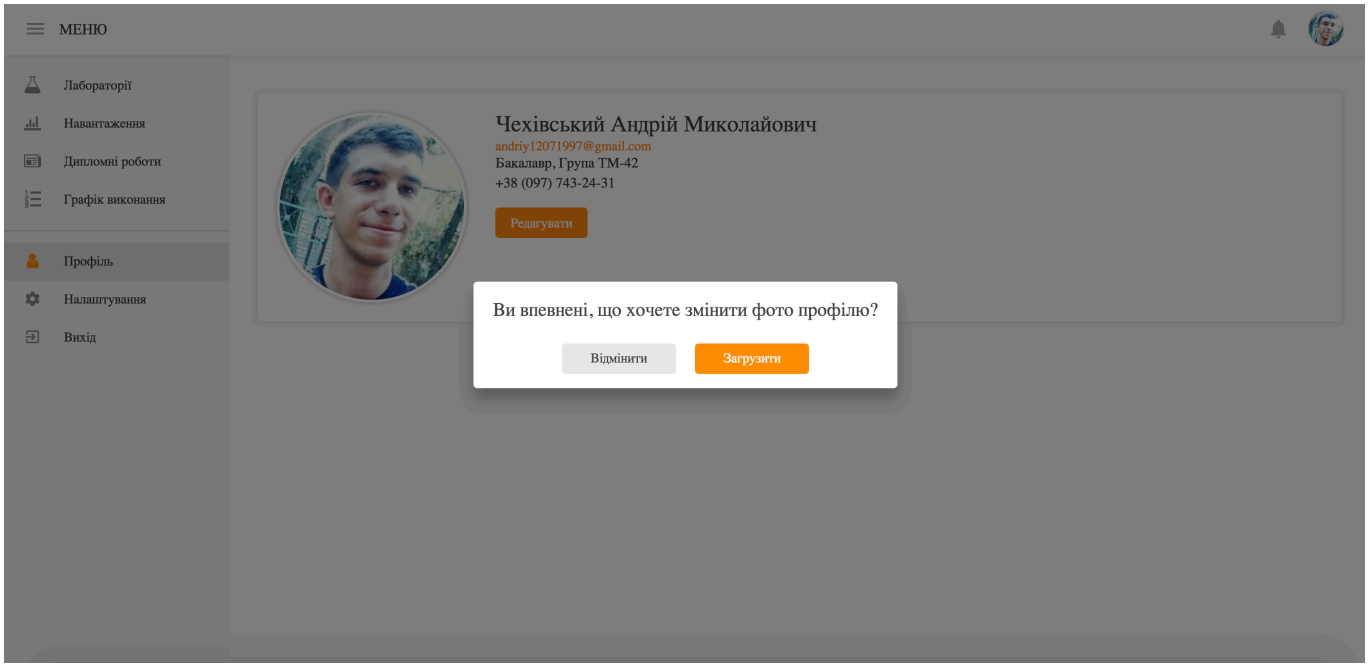

Рисунок 5.9 — Зразок інтерфейсу вибору фото для зміни аватара користувача

Після вибору фотографії відкимається модальне вікно в якому користувач може модифікувати аватар і також зберегти його (рисунок 5.10).

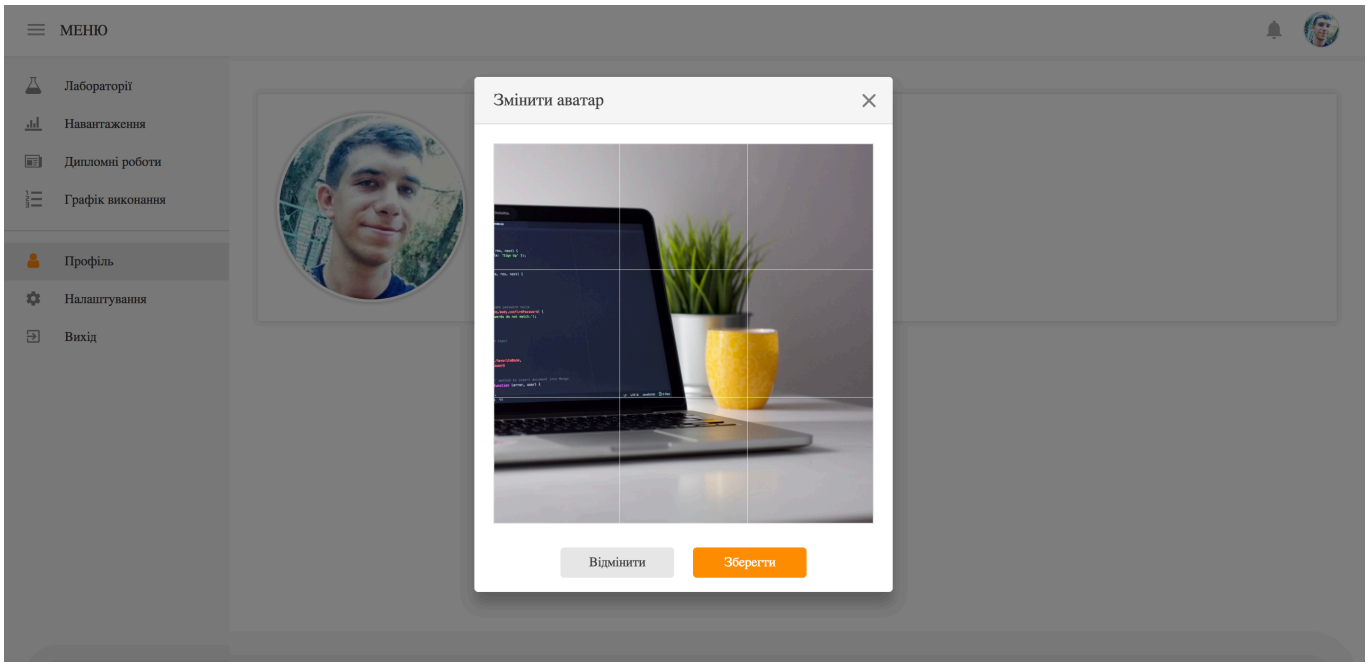

Рисунок 5.10 — Зразок інтерфейсу зміни аватара користувача

Студент має можливість змінювати свою електронну адресу, номер телефону, а також кольорову тему застосунку (рисунок 5.11).

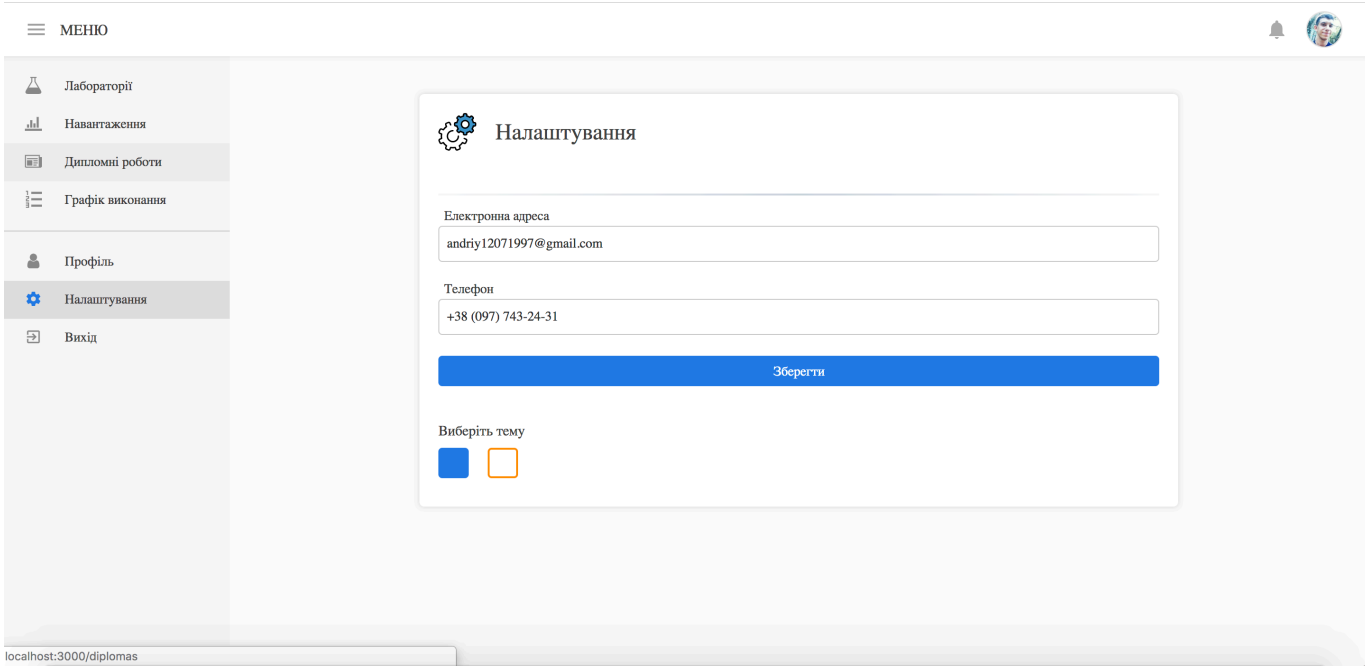

Рисунок 5.11 — Зразок інтерфейсу налаштувань користувача

Студент має можливість переглядати лабораторії (рисунок 5.12).

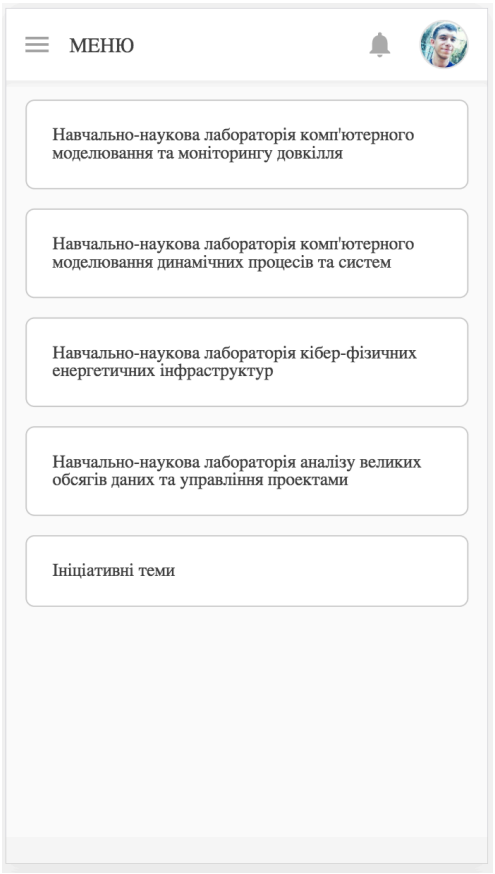

Рисунок 5.12 — Зразок інтерфейсу лабораторій

Студент має можливість переглядати інформацію про напрямки лабораторії і слад лабораторії (рисунок 5.13).

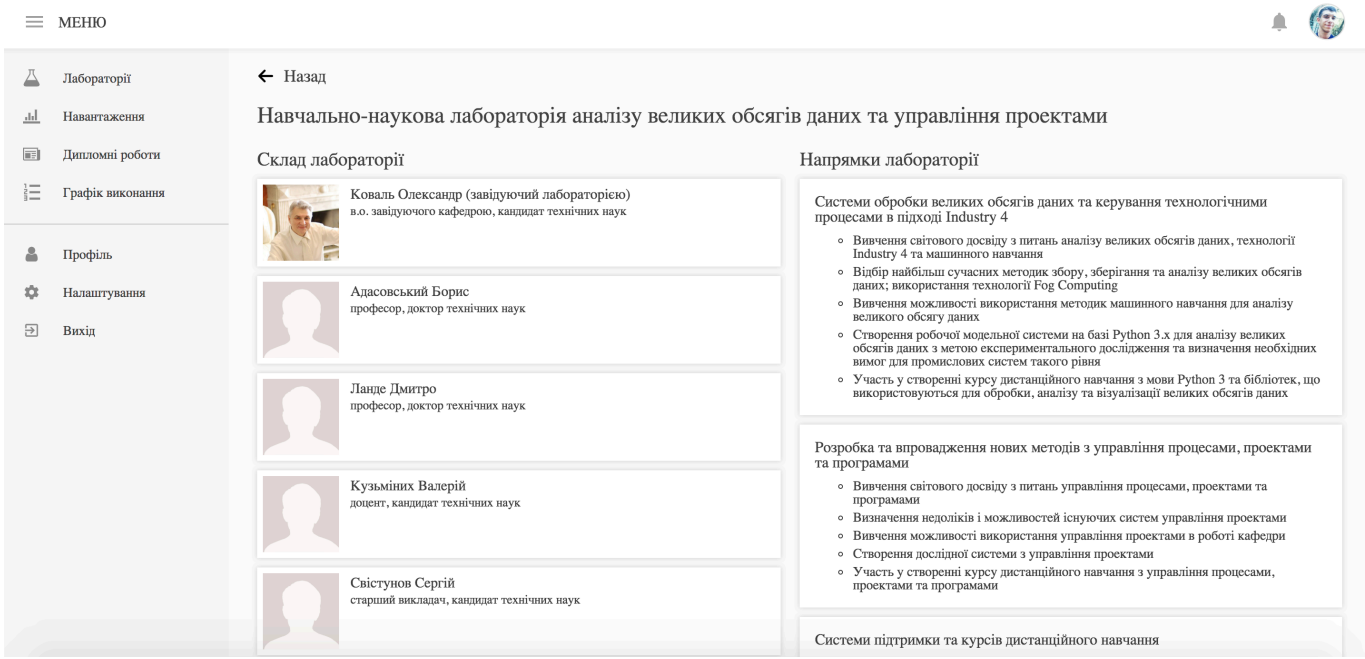

Рисунок 5.13 — Зразок інтерфейсу напрямків та складу лабораторії

При натисканні на викладача студент перенаправляється на дипломні роботи, де в фільтрах буде знаходить вибраний викладач (рисунок 5.14).

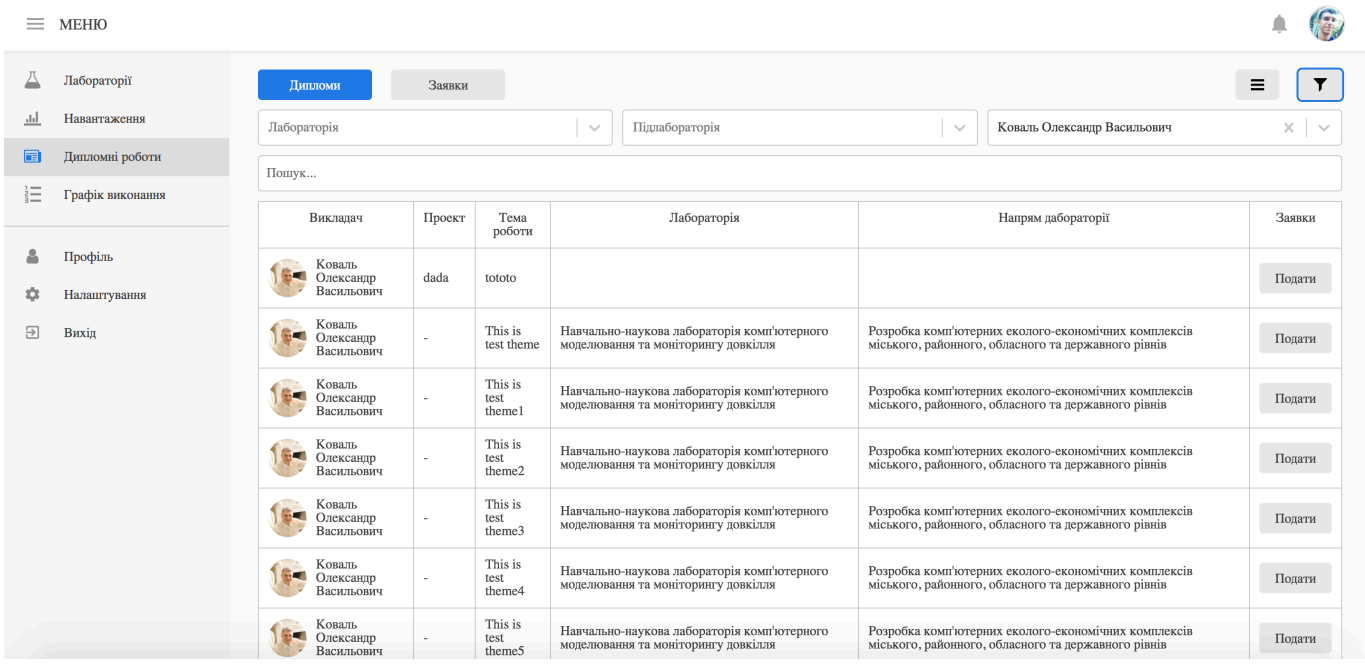

Рисунок 5.14 — Зразок інтерфейсу фільтрування тем дипломних робіт

Студент має можливість переглядати навантаження для викладачів (рисунок 5.15).

Рисунок 5.15 — Зразок інтерфейсу навантаження на викладачів

Студент має можливість переглядати графік виконання дипломної роботи (рисунок 5.16).

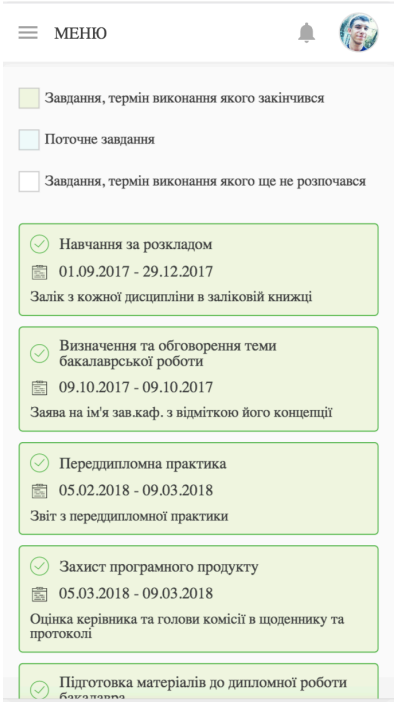

Рисунок 5.16 — Зразок інтерфейсу графіку виконання дипломної роботи

Також в застосунку є контексне меню. Для того, щоб його відкрити потрібно натиснути на аватар користувача в правому верхньому куті (рисунок 5.17).

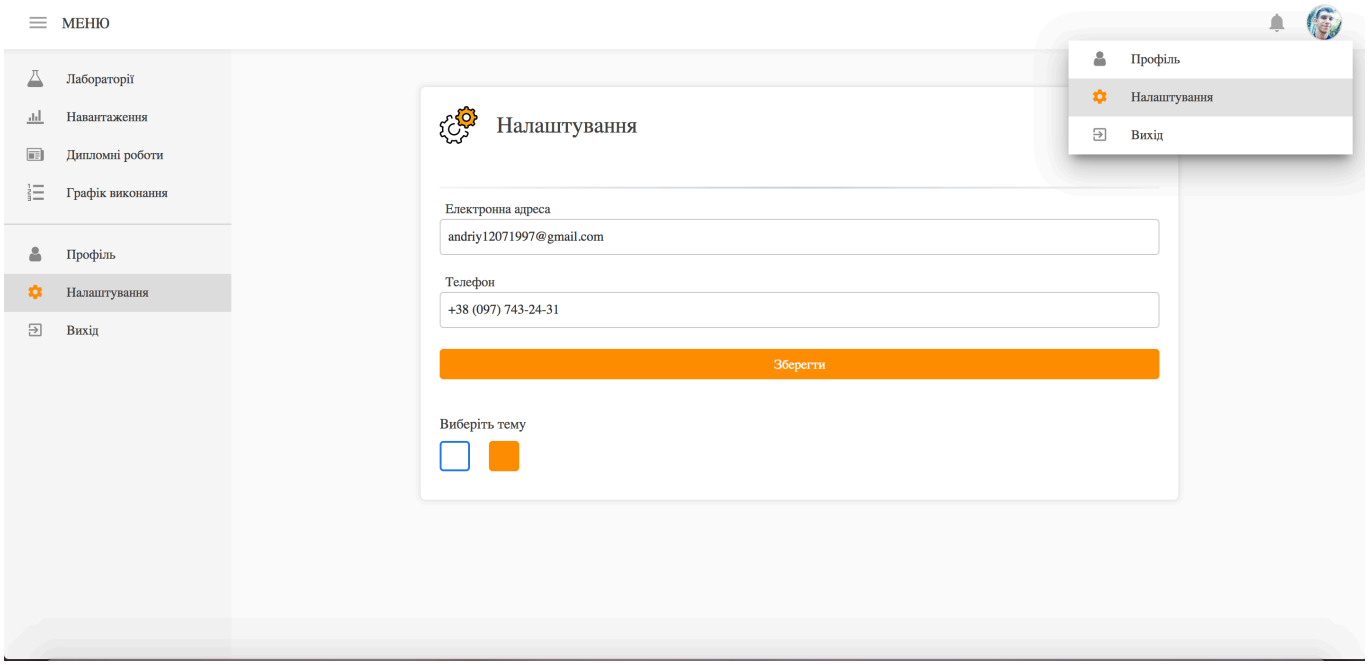

Рисунок 5.17 — Зразок інтерфейсу контекстного меню

Також бокове меню може скриватися, а на мобільних пристроях виїжджати збоку (рисунок 5.18).

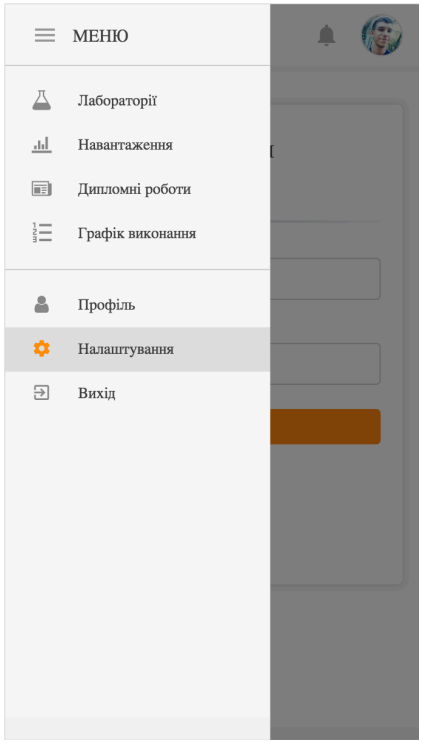

Рисунок 5.18 — Зразок інтерфейсу бокового меню

5.1.Технічні вимоги до середовищ використання

Мінімальними вимогами для процессорів, які запускаються в браузер на операційній системі Windows є Internet Explorer 8. Він потребує 233 Mhz процесор; мінімум 64 MB RAM (для Windows XP), рекомендовано 512 MB RAM; мінімум 150 MB (Windows XP), 70 MB (Windows Vista) вільного місця на диску.

5.2.Загальний опис роботи веб-застосунку

Технологія використання застосунку полягає в наступному: корисувач виконує вхід в систему, після чого корисувач бачить список дипломних робіт, на які він має можливість подати заявку. Після подачі заявки, користувач очікує коли викладач прийме заявку користувача. Після прийняття заявки користувач бачить інформацію про дипломну роботу.

ВИСНОВКИ

В результаті виконаної роботи було розроблено систему, яка надає можливість шукати та переглядати теми дипломних робіт, подавати і відміняти заявки на дипломні роботи, переглядати інформацію про свою дипломну роботу після прийняття заявки викладачем, графік виконання дипломної роботи, інформацію про склад та напрямки лабораторії, навантаження на викладачів.

Проаналізовано існуючі системи управління проектами, визначенно їх переваги та недоліки. Проаналізовано вимоги до нової системи управління дипломними проектами студента.

Базу даних розроблений за допомогою системи керування базами даних MySQL. Її будова відповідає функціям та вимогам розроблюваної системи.

З використанням бібліотеки React.js та мови програмування JavaScript реалізовано графічний веб-застосунок для управління дипромними проектами.

Для зв’язку з базою даних та взаємодії з клієнтами за допомогою API було створено Node.js сервер.

Завдяки тому, що програмна система написана за допомогою сучасної мови програмування та її провідних бібліотек, гарантується з часом підтримка всіх компонентів та їх подальша модернізація.

Дана система значно спрощує процес управління дипломними проектами.

Реалізовані можливості програмного продукту повністю задовільняють поставленій задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Умберто Е., Як написати дипломну роботу. Пер. за ред. О. Глотова. — Тернопіль: Мандрівець, 2007. — 224 с.

2. SOAP and XML-RPC API Deprecation Notice [Електронний ресурс] — 2017. — Режим доступу до ресурсу: https://developer.atlassian.com/server/jira/platform/soap-and-xml-rpc-api-deprecation-notice/

3. Choosing a default language [Електронний ресурс] — Режим доступу до ресурсу: https://confluence.atlassian.com/adminjiraserver071/choosing-a-default-language-802592304.html

4. Эффективное программирование TCP/IP / Йон Снейдер Й – ДМК Пресс, 2009 — 320 с.

5. OCLC [Електронний ресурс]. — 2017. — Режим доступу до ресурсу: https://www.oclc.org/en/home.html?redirect=true.

6. Дэвид Флэнаган. JavaScript. Подробное руководство, 6-е издание, 2012, 1080 с.

7. Выразительный Javascript [Электронный ресурс] — Режим доступа: https://karmazzin.gitbooks.io/eloquentjavascript_ru/content/

8. Резиг Джон, Бибо Беэр. Секреты javascript ниндзя. — М.: «Вильямс», 2013. — 416 с.

9. Марина Дмитриева. JavaScript. — С-П.: БХВ-Петербург, 2004. — 336 с.

10. Современные возможности ES-2015 [Электронный ресурс] — Режим доступа: https://learn.javascript.ru/es-moder

11.Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. — O'Reilly Media, 2017. — 350 p.

12. Sidelnikov G. React.js Book: Learning React JavaScript Library From Scratch. — River Tigris LLC, 2016. — 350 p.

13. Bertolli M. React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns. — Packt Publishing, 2017. — 320 p.

14. Carlos R. React Cookbook: Create dynamic web apps with React using Redux, Webpack, Node.js, and GraphQL. — Packt Publishing, 2018. — 580 p.

15. Gorgon Z. React Explained: Your Step-by-Step Guide to React. – Amazon Digital Services LLC, 2019. — 305 p.

16. Wieruch R. The Road to learn React: Your journey to master plain yet pragmatic React.js. — Amazon Digital Services LLC, 2017. — 202 p.

17. Гибкая разработка веб-приложений / Сэм Руби., 2014. — 448 с. — (Для профессионалов).

18. The MIT License (MIT) [Електронний ресурс] — Режим доступу до ресурсу: https://opensource.org/licenses/MIT.

19. Стив Макконнелл. Совершенный код = Code complete. — СПб.: Питер, 2005. — С. 896.

20. Большая книга CSS3 [Электронный ресурс] — Режим доступа:ftp://ftp.micronet-rostov.ru/linux-support/books/programming/HTML-CSS/Дэвид Сойер Макфарланд — Большая книга CSS3.pdf.

21. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. — М. : Вильямс, 2003. — 1440 с.

22. What is REST [Electronic resource]: — Access mode: http://www.restapitutorial.com/lessons/whatisrest.html.

23. Design a beautiful REST API [Electronic resource]: — Access mode: https://medium.com/@zwacky/design-a-beautiful-rest-api-901c73489458

24. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / И. Браун — Санкт-Петербург: Питер, 2017. — 336 с

25. Pereira C. Node.js Building APIs with Node.js. — Apress, 2016. — 136 p.

ДОДАТОК А

Модуль “Студент” системи управління дипломними проектами.

Специфікація

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51164_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51164_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51164_19Б 12-1	App.js Layout/index.js Modal/index.js	Основні компоненти
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51164_19Б 13-1	Додаток 3.doc	Опис програмного модуля

ДОДАТОК Б

Модуль “Студент” системи управління дипломними проектами.

Текст програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51164_19Б 12-1

Аркушів 7

Київ 2019

```
// App.js
/* REACT */
import React from 'react'

/* MODULES */
import { Switch, Router, Route, Redirect } from 'react-router-dom'
import { connect } from 'react-redux'
import { ThemeProvider } from 'styled-components'

/* CUSTOM MODULES */
import RouteWithSubRoutes from '/src/routes/routeWithSubRoutes'
import { Layout } from '/src/components/HOC'
import history from './utils/history'
import { AuthService } from '/src/utils'
import { Loader } from '/src/components/UI'
import GlobalLoader from '/src/components/GlobalLoader'
import GlobalStyle from '/src/styles/global'
import { NotificationProvider } from './components/HOC'
import AuthPage from './containers/Auth'
import ForgotPasswordPage from '/src/containers/Auth/ForgotPassword'
import { routesSelector } from '/src/store/selectors/authSelectors'

/* ACTIONS */
import { loginAction, tryAutoLogin, loadedApp } from './store/actions/
authActions'
import { referenceAction } from './store/actions/referenceActions'

/* STYLES */
import { getTheme } from './styles/theme'

@connect(
  state => ({
    auth: state.auth,
    settings: state.settings,
    ROUTES: routesSelector(state)
  }),
  { loginAction, tryAutoLogin, loadedApp, referenceAction }
)
class App extends React.Component {
  componentDidMount () {
```

```
    const { tryAutoLogin, referenceAction, loadedApp } = this.props

    if (AuthService.getCredentials()) {
      tryAutoLogin(() => { referenceAction() })
    } else {
      loadedApp()
    }
  }

  render () {
    const { auth, settings, ROUTES } = this.props
    const theme = getTheme(settings.theme)
    // Can use different switches or private route and login route for protect
routes
    let routes = (
      <Switch>
        {(
          ROUTES && ROUTES.map(route => (
            <RouteWithSubRoutes
              key={route.path}
              {...route}
            />
          ))
        )}
        <Redirect to='/' />
      </Switch>)

    if (!auth.token) {
      routes = (<Switch>
        <Route path='/login' component={AuthPage} exact />
        <Route path='/forgot-password' component={ForgotPasswordPage} exact />
        <Redirect to='/login' />
      </Switch>)
    }

    let content = (
      <Layout>
        {routes}
      </Layout>
    )
```

```
        if (!auth.isLoaded) content = (<Loader size='3rem'>Авторизація...</Loader>)

        return (
          <Router history={history}>
            <ThemeProvider theme={theme}>
              <NotificationProvider>
                <GlobalStyle />
                {content}
                                {auth.isLoaded && settings.isGlobalLoading &&
<GlobalLoader>Завантаження...</GlobalLoader>}
              </NotificationProvider>
            </ThemeProvider>
          </Router>
        )
      }
    }
    export default App


    // Layout/index.js
    import React, { Component, Fragment } from 'react'
    import { connect } from 'react-redux'

    import Header from 'src/components/Header'
    import SideDrawer from 'src/components/Navigation/SideDrawer'

    import history from '/src/utils/history'
    import { MainContainer } from './style'
    import { settingsAction, toggleSideDrawerAction } from '/src/store/actions/
settingsActions'
    import { isMobileSelector } from '/src/store/selectors/settingsSelectors'
    import { navigationItemsSelector } from '/src/store/selectors/authSelectors'

    @connect(
      state => ({
        settings: state.settings,
        auth: state.auth,
        isMobile: isMobileSelector(state),
        navigationItems: navigationItemsSelector(state)
      }),
```

```
        { settingsAction, toggleSideDrawerAction },
        null,
        { pure: false }
      )
      class Layout extends Component {
        componentWillMount () {
          const clientWidth = document.documentElement.clientWidth
           this.props.settingsAction({ clientWidth, isOpenSideDrawer: clientWidth >=
1000 })
        }

        componentDidMount () {
          window.addEventListener('resize', this.updateViewState)
          history.listen((location) => {
              if (location.pathname === '/login' || location.pathname === '/forgot-
password') {
              if (this.props.auth.token) {
                history.go(1)
              }
            } else if (!this.props.auth.token) {
              history.push('/login')
            }
            this.props.settingsAction({ isShowAnimation: false })
          })
        }

        componentWillUnmount () {
          window.removeEventListener('resize', this.updateViewState)
        }

        updateViewState = () => {
            this.props.settingsAction({clientWidth:
document.documentElement.clientWidth })
        }

        render () {
          const {
            settings: { isOpenSideDrawer, isShowAnimation, isMobile },
            navigationItems,
            auth: { token },
```

```
      toggleSideDrawerAction,
      children
    } = this.props
    const isLoginPage = history.location.pathname === '/login' || !token

    return (
      <Fragment>
        {!isLoginPage && (
          <Fragment>
            <Header isMobile={isMobile} />
            <SideDrawer
              isOpen={isOpenSideDrawer}
              isShowAnimation={isShowAnimation}
              toggleSideDrawer={toggleSideDrawerAction}
              navigationItems={navigationItems}
            />
          </Fragment>
        )}
        <MainContainer
          isLoginPage={isLoginPage}
          isOpenSideDrawer={isOpenSideDrawer}
          isShowAnimation={isShowAnimation}
        >
          {children}
        </MainContainer>
      </Fragment>
    )
  }
}
export default Layout

// commonSaga.js
import { call, put, takeEvery } from 'redux-saga/effects'

import sagasConfig from './config'
import { ApiCaller, AuthService } from '/src/utils'

const successStatuses = [200, 201, 204]

function * processRequest ({
```

```
  type, payload, onSuccess, onFail
}) {
  const {
    path,
    method,
    requiresAuth,
    successType,
    failType,
    multipart
  } = sagasConfig[type]
  let token = null
  let role = null
  if (requiresAuth) {
    ({ token, role } = yield call(AuthService.getCredentials))
  }

  const url = yield call(path, payload)
  const response = yield call(
    ApiCaller, url, method, payload, token, role, multipart
  )

  if (successStatuses.includes(response.status)) {
    if (response.status === 204) {
      yield put({ type: successType, payload })
    } else {
      yield put({ type: successType, payload: response.data })
    }
    if (typeof onSuccess === 'function') {
      yield call(onSuccess, response.data)
    }
  } else {
    if (response.status === 401) {
      AuthService.unsetCredentials()
      window.location.reload()
    }
    yield put({ type: failType, payload: response.data })
    if (typeof onFail === 'function') {
      yield call(onFail, response.data)
    }
  }
```

```
}

export const commonSaga = function * () {
  yield takeEvery(
    ({ type }) => /(.*)_REQUEST/.test(type) &&
      !sagasConfig.ignoreTypes.includes(type),
    processRequest
  )
}


export default Modal
```

ДОДАТОК В

Модуль “Студент” системи управління дипломними проектами.

Опис програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51164_19Б 13-1

Аркушів 8

Київ 2019

АНОТАЦІЯ

Розроблений програмний продукт представляє собою веб-застосунок і дозволяє користувачам шукати та переглядати теми дипломних робіт, подавати і відміняти заявки на теми дипломних робіт, переглядати інформацію по своїй дипломній роботі, переглядати навантаження на викладачів. Модуль написано мовою програмування JavaScript, з використанням бібліотеки React.js, бібліотеки керування станом системи Redux.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Для роботи програми необхідно встановити платформу Node.js. Разом з нею встановлюється менеджер пакетів npm. Після цього потрібно встановити create-react-app. Для цього виконуємо команди “npm install -g create-react-app”.

Для запуску додатку потрібно відкрити проект у WebStorm або відкрити в терміналі директорію з проектом і виконати наступні команди: “npm install && npm start”.

Основний модуль був написаний на мові JavaScipt за допомогою бібліотеки React у середовищі розробки Webstorm.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система має функції пошуку та перегляду тем дипломних робіт, подачі та відміни заявок на теми дипломних робіт, перегляду інформації по своїй дипломній роботі після прийняття заявки викладачем, перегляду графіка виконання дипломної роботи, навантаження на викладачів, інформації про склад та напрямки лабораторії, редагування та перегляду профілю, входу в систему, відновлення паролю.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Логічна структура програми складається з класів та функції, які виконують поставлені задачі. Програма містить директорію компонентів, де знаходяться спільні компоненти для користувацького інтерфейсу. За стилізацію додатку відповідає директорія стилів. За маршрутизацію додатку відповідають модулі в директорії шляхів. За збереження даних та їх зміну відповідають модулі сховищ, які знаходяться в директорії сховищ. Також існують допоміжні модулі, які формують структуру програми — це конфіги, допоміжні функції тощо.

Усі модулі програми є незалежними та здатні до масштабування. Саме завдяки правильній побудові структури програми, вона є гнучкою, легкою в подальшій розробці та підтримці.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модулі розроблено у середовищі розробки Webstorm, на комп’ютері, що використовував операційну систему macOS High Sierra. Мовою програмування було обрано JavaScript та бібліотеку React. Мобільний додаток написаний для операційної системи macOS, Windows та Linux.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску веб-застосунку потрібно відкрити будь-який браузер в операційній системи macOS, Windows або Linux.

Для використання даного модулю не потрібно ніяких дій, оскільки він автоматично спрацьовує після запуску клієнтського додатку.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є інформація, яку користувач вводить в застосунку.

Вихідними даними є списки тем дипломних робіт, заявок, навантаження, графік виконання, а також сторінки профілю і лабораторій.